Speech Recognition for Robotic Control

Shafkat Kibria

December 18, 2005
Master's Thesis in Computing Science, 20 credits
Supervisor at CS-UmU: Thomas Hellström
Examiner: Per Lindström

Umeå University
Department of Computing Science
SE-901 87 UMEÅ
SWEDEN

Abstract

The term "robot" generally connotes some anthropomorphic (human-like) appearance [24]. Brooks [5] research coined some research issues for developing humanoid robot and one of the significant research issues is to develop machine that have human-like perception. What is human-like perception? - The five classical human sensors - vision, hearing, touch, smell and taste; by which they percept the surrounding world. The main goal of our project is to introduce "hearing" sensor and also the speech synthesis to the Mobile robot such that it is capable to interact with human through Spoken Natural Language (NL). Speech recognition (SR) is a prominent technology, which helps us to introduce "hearing" as well as Natural Language (NL) interface through Speech for the Human-Robot interaction. So the promise of *anthropomorphic* robot is starting to become a reality. We have chosen Mobile Robot, because this type of robot is getting popular as a service robot in the social context, where the main challenge is to interact with human. Two type of approach we have chosen for Voice User Interface (VUI) implementation - using a Hardware SR system and another one, using a Software SR system. We have followed Hybrid architecture for the general robotics design and communication with the SR system; also created the grammar for the speech, which is chosen for the robotic activities in his arena. The design and both implementation approaches are presented in this report. One of the important goals of our project is to introduce suitable user interface for novice user and our test plan is designed according to achieve our project goals; so we have also conducted a usability evaluation of our system through novice users. We have performed tests with simple and complex sentences for different types of robotics activities; and also analyzed the test result to find-out the problems and limitations. This report presents all the test results and the findings, which we have achieved through out the project.

Contents

1 Introduction **1**

2 Literature Review **3**

2.1 About Robot 3

2.2 Speech Recognition 4

2.3 VUI (Voice user interface) in Robotics 9

3 Language and Speech **11**

3.1 Speech 12

3.1.1 Speech Synthesis 12

3.1.2 Speech Recognition System 12

3.2 Grammar 12

4 Implementation **15**

4.1 General Robotic Design 16

4.1.1 Behaviors' Algorithm 19

4.2 Hardware Approach 21

4.2.1 System Component 22

4.2.2 System Design 23

4.2.3 Algorithm Description 27

4.3 Software Approach 29

4.3.1 System Component 30

4.3.2 System Design 32

4.3.3 Algorithm Description 33

5 Evaluation **35**

5.1 Test Plan 35

5.2 Results 36

5.2.1 Hardware approach 36

5.2.2 Software approach 36

5.2.3 Experience from the Technical Fair 36

6 Discussion **45**

7 Conclusions **47**
7.1 Limitations 48
7.2 Future work 48

8 Acknowledgements **49**

References **51**

A Hardware & Software Components **55**
A.1 Hardware Components 55
A.1.1 Voice $Extreme^{TM}$ (VE) Module 55
A.1.2 Voice $Extreme^{TM}$ (VE) Development Board 56
A.1.3 Khepera 57
A.2 Software Components 58
A.2.1 Voice $Extreme^{TM}$ IDE 58
A.2.2 SpeechStudio 59

B Installation guide **61**
B.1 Developer guide 61
B.1.1 Speech Recognition software product installation 61
B.1.2 The Source code files 62
B.2 User guide 62

C User Questionnaire **65**

D Glossary **67**

List of Figures

2.1 Three paradigms a) Hierachical b) Reactive c) Hybrid deliverative/reactive [24]. . . . 4
2.2 Typical Spoken Natural Language Interface in Robotic. . . . 10

3.1 A context-free grammar for simple expressions (i.e., *a+b or ab+ba* etc.) 13

4.1 Hybrid architecture for our prototype 19
4.2 Forward kinematics for the Khepera Robot [15] 20
4.3 The robot can able to handle this kind of situations through Bug algorithm [14]. . . . 20
4.4 Overview of Hardware approach system. . . . 21
4.5 The circuit diagram of the interface between Khepera General I/O Turret and VE Module 24
4.6 The picture of Khepera with VE Module. . . . 25
4.7 Command-Sentence-Packet's Structure. . . . 25
4.8 The Grammar for the language model. . . . 26
4.9 The Design for Semantic Analysis. . . . 27
4.10 Overview of Software approach system. . . . 30
4.11 An overview picture of interfacing SpeechStudio SR system with VB6.0 [35]. . . . 32
4.12 An example of "Option Button" and "Text Box" use for "Move" and "Turn" behaviors. . . . 32
4.13 An example of create grammar to activate "Option Button" and to send parameter at "Text Box" for "Turn" behavior. . . . 33

5.1 The picture of the CARO's arena (outside view) 37
5.2 The picture of the CARO's arena (inside view) 38
5.3 Curious visitors are watching the CARO (The picture from the Technical fair) 38
5.4 The histogram shows the user's information on the basis of age and sex. 40

5.5 The histogram shows participant user's information on the basis of age and occupation. 40
5.6 The user comments about controlling the CARO. 41
5.7 The Users comment about CARO's efficiency. 42
5.8 The Users Comment about flexibility. 43
5.9 The Users comment about their preferences. 44

A.1 Voice $Extreme^{TM}$ (VE) Module [31]. 55
A.2 Voice $Extreme^{TM}$ (VE) Module's Pins Configuration [31]. 56
A.3 Voice $Extreme^{TM}$ (VE) Development Board [32]. 56
A.4 Voice $Extreme^{TM}$ (VE) Development Board I/O pins configuration [32]. 57
A.5 Khepera (a small mobile robot) [18]. 57
A.6 Overview of the GENERAL I/O TURRET [18]. 58
A.7 Voice $Extreme^{TM}$ IDE [32]. 59
A.8 SpeechStudio workspace window. 60
A.9 SpeechStudio grammar creation environment for developer. 60

List of Tables

2.1 Speech Recognition Techniques [7]. 5
2.2 Languages Support by the available Speech Recognition Software Program [12]. 6
2.2 (Continued) [12]. 7
2.3 Some of the available *SR* programs for developer and their vendors. . . 8
2.4 Some of the Available SR Hardware Module and their Manufacturer. . . 8

4.1 Simple Sentences for robotic activities. 15
4.2 Simple Sentences for some complex robotic activities. 16
4.3 Complex Sentences for robotic activities. 16
4.4 The behaviors identified for the prototype degin. 17
4.5 The summary of Hybrid architecture (Figure 4.1) in terms of the common components and style of emergent behavior. 18
4.6 The Lexicon for the language model. 26

B.1 The available software products and their file's name in the SpeechStudio Developer Bundle Package. 61

Chapter 1

Introduction

The theme of Social interaction and intelligence is important and interesting to an Artificial intelligence and Robotics community [9]. It is one of the challenging areas in Human-Robot Interaction (HRI). Speech recognition technology is a great aid to admit the challenge and it is a prominent technology for Human-Computer Interaction (HCI) and Human-Robot Interaction (HRI) for the future.

Humans are used to interact with Natural Language (NL) in the social context. This idea leads Roboticist to make NL interface through Speech for the HRI. Natural Language (NL) interface is now starting to appear in standard software application. This gives benefit to novices to easily interact with the standard software in HCI field. Its also encourage Roboticist to use Speech Recognition (SR) technology for the HRI. To percept the world is important knowledge for the knowledge Based-Agent and Robot to do a task. It's also a key factor to know initial knowledge about the Unknown world. In the social context Robot can easily interact with Human through SR to gain the initial knowledge about the Unknown world and also the information about the task to accomplish.

There are several *SR* interface robotic systems have been presented [30, 6, 22, 20, 11, 17]. Most of the projects emphasize on Mobile Robot - now a days this type of robot is getting popular as a service robot at indoor and outdoor[1]. The goal of the service robot is to help people in everyday life at social context. It is an important thing for the Mobile robot to communicate with the users (human) of its world. So Speech Recognition (SR) is an easy way of communication with human and it also gives the advantage of interacting with the novice users without a proper training. Uncertainty is a major problem for navigation systems in mobile robots - interaction with humans in a natural way, using English rather than a programming language, would be a means of overcoming difficulties with localization. [30]

In this project our main target is to add *SR* capabilities in the Mobile Robot and investigate the use of a natural language (NL) such as English as a user interface for interacting with the Robot. We choose small Mobile Robot (Khepera) for this investigation. We try both with hardware Speech Recognition (SR) device and as well as Software PC based SR to achieve our goal. Both technologies are used for *SR* system

[1] World Robotics survey 2004 - issued by UNECE: United Nations Economic Commission for Europe.

depending on the vocabulary size and the complexity of the grammar. We define several requirements for our prototype system. Interaction with robot should be in natural spoken English (within the application domain). We choose English, because it is most recognized international Language. The robot should understand its task from the dialogues has spoken. The system should be user independent.

In the following chapter we are going to discuss more about the *SR* system and most important parts - introducing *SR* system to the Robotic for interaction purpose. We start with the literature review about *SR* system and Voice User Interface (VUI) system (Chapter 2 on page 3). Then we discuss about the important components of Language and Speech in Chapter 3 (on page 11). This includes Speech, Speech synthesizer, Speech Recognition Grammar etc. Chapter 4 (on page 15) contains the description about the implementation part of our project. There, we discuss about the components -we used for implementation the system and also the mechanism of the system. Later on, we have presented our test result in the Chapter 5 (on page 35) and we also do a discussion about the result - we have presented (see in Chapter 6 on page 45). We conclude in Chapter 7 (on page 47), in the conclusion part we discuss about the limitation as well as future work.

Chapter 2

Literature Review

Worldwide investment in industrial robots up 19% in 2003. In first half of 2004, orders for robots were up another 18% to the highest level ever recorded. Worldwide growth in the period 2004-2007 forecast at an average annual rate of about 7%. Over 600,000 household robots in use - several millions in the next few years.
UNECE issues its 2004 World Robotics survey [36]

From the above press release we can easily realize that household (service) robots getting popular. This gives the researcher more interest to work with service robots to make it more user friendly to the social context. Speech Recognition (SR) technology gives the researcher the opportunity to add Natural language (NL) communication with robot in natural and even way in the social context. So the promise of robot that behave more similar to humans (at least from the perception-response point of view) is starting to become a reality [28]. Brooks research [5] is also an example of developing humanoid robot and raised some research issues. Form these issues; one of the important issues is to develop machine that have human-like perception.

2.1 About Robot

The term "robot" generally connotes some anthropomorphic (human-like) appearance; consider robot "arms" for welding [24]. The main goal robotic is to make Robot workers, which can smart enough to replace human from labor work or any kind of dangerous task that can be harmful for human. The idea of robot made up mechanical parts came from the science fiction. *Three classical films,* Metropolis *(1926),* The Day the Earth Stood Still *(1951), and* Forbidden Planet *(1956), cemented the connotation that robots were mechanical in origin, ignoring the biological origins in Capek's play*[24]. To work as a replacement of human robot need some Intelligence to do function autonomously. AI (Artificial intelligence) gives us the opportunity to fulfill the intelligent requirement in robotics. There are three paradigms are followed in AI robotics depends on the problems. These are - Hierarchical, Reactive, and Hybrid deliberative/reactive. *Applying the right paradigm makes problem solving easier* [24]. Depending on three commonly accepted robotic primitives the overview of three paradigms of robotics on Figure 2.1.

In our project we follow Hybrid deliberrative/reactive paradigm to slove our robotic problem. (See detail in Chapter 4 on page 15).

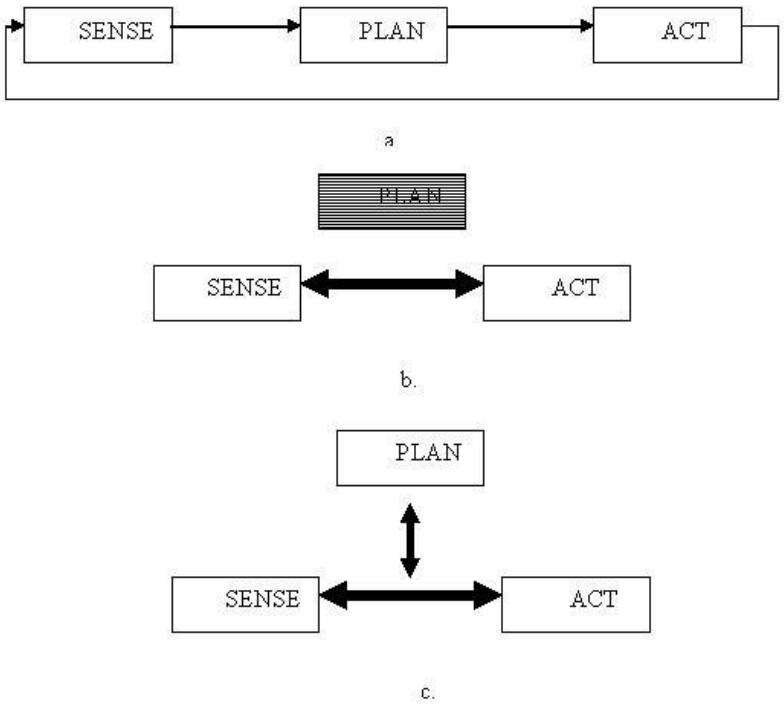

Figure 2.1: Three paradigms a) Hierachical b) Reactive c) Hybrid deliverative/reactive [24].

2.2 Speech Recognition

Speech Recognition technology promises to change the way we interact with machines (robots, computers etc.) in the future. This technology is getting matured day by day and scientists are still working hard to overcome the remaining limitation. Now a days it is introducing many important areas (like - *in the field of* Aerospace *where the training and operational demands on the crew have significantly increased with the proliferation of technology* [27], in the *Operation Theater* as a surgeon's aid to control lights, cameras, pumps and equipment by simple voice commands [1]) in the social context.

Speech recognition is the process of converting an acoustic signal, captured by microphone or a telephone, to a set of words [8]. There two important part of in Speech Recognition - i) Recognize the series of sound and ii) Identified the word from the sound. This recognition technique depends also on many parameters - Speaking Mode, Speaking Style, Speaker Enrollment, Size of the Vocabulary, Language Model, Perplexity, Transducer etc [8]. There are two types of Speak Mode for speech recognition system - one word at a time (isolated-word speech) and continuous speech. Depending on the speaker enrolment, the speech recognition system can also divide - Speaker dependent and Speaker independent system. In Speaker dependent systems user need to be train the systems before using them, on the other hand Speaker independent system can identify any speaker's speech.Vocabulary size and the language model also important

factors in a Speech recognition system. Language model or artificial grammars are used to confine word combination in a series of word or sound. The size of the vocabulary also should be in a suitable number. Large numbers of vocabularies or many similar-sounding words make recognition difficult for the system.

The most popular and dominated technique in last two decade is Hidden Markov Models. There are other techniques also use for SR system - Artificial Neural Network (ANN), Back Propagation Algorithm (BPA), Fast Fourier Transform (FFT), Learn Vector Quantization (LVQ), Neural Network (NN) [7].

Techinque	**Sub Technique**	**Relevant Variable(s)/Data Structures**	**Input**	**Output**
Sound Sampling	ALL	Analog Sound Signal	Analog Sound Signal	Digital Sound Samples
Feature Extraction	Dynamic Time Warping (DTW)	Statistical Features (e.g. LPC coefficients)	Digital Sound Samples	Acoustic Sequence Templates
	Hidden Markov Models (HMM)	Subword Features (e.g. phonemes)	Digital Sound Samples	Subword Features (e.g. phonemes)
	Artificial Neural Networks (ANN)	Statistical Features (e.g. LPC coefficients)	Digital Sound Samples	Statistical Features (e.g. LPC coefficients)
Training and Testing	Dynamic Time Warping (DTW)	Reference Model Database	Acoustic Sequence Templates	Comparison Score
	Hidden Markov Models (HMM)	Markov Chain	Subword Features (e.g. phonemes)	Comparison Score
	Artificial Neural Networks (ANN)	Neural Network with Weights	Statistical Features (e.g. LPC coefficients)	Positive/ Negative Output

Table 2.1: Speech Recognition Techniques [7].

There are both Speech Recognition Software Program (SRSP) and Speech Recognition Hardware Module (SRHM) is available now in the market. The *SRSP*s are more mature then *SRHM*s, but it is available for limited number of languages [12]. See Table 2.2 - A complete list of available languages for Speech Recognition Software Program (SRSP). Table 2.3 shows the available *SR* programs for developer and their vendors.

Language	DNS Preferred Versions 7 & 8	Microsoft SR (Office 2003)	ViaVoice Version 10	Other applications
Arabic	NO	NO	[Last version was Millenium / 7, but it has disappeared]	
Catalan	NO	NO	NO	Was available from Philips FreeSpeech 2000 (only Windows only up to 98), but discontinued
Chinese	NO	YES	NO	
Dutch	YES(Package also includes full English, French, and German)	NO	No longer mentioned on ScanSoft Website	
English	YES - US, UK, Australian, SE Asian (all in one package). Latest version - 8. Same collection also available as a component of packages in all other languages. Latest version - 7.1	US (but easily accommodates other varieties, though only US spelling available)	US, UK (used to be sold separately)	
French	YES (Package also includes full English)	NO	No longer mentioned on ScanSoft Website	
German	YES (Package also includes full English) Latest version - 8.	NO	YES	

Table 2.2: Languages Support by the available Speech Recognition Software Program [12].

Table 2.2: (Continued) [12].

Language	DNS Preferred Versions 7 & 8	Microsoft SR (Office 2003)	ViaVoice Version 10	Other applications
Italian	YES (Package also includes full English) Latest version - 8.	NO	YES	
Japanese	YES	YES	YES	
Portuguese	NO	NO	Latest version: 9 for Brazilian only; No longer mentioned on ScanSoft Website, but still available from some stores.	
Spanish	YES (Package also includes full English)	NO	No longer mentioned on ScanSoft Website	
Swedish	NO	NO	NO	Available from Voxit, Stockholm (VoiceXpress, latest version: 5.2)
Multilingualism	Version 7 Supports all available languages Version 8 Does NOT support all languages, only those included in a package	Not applicable	Supports all available languages	Philips FreeSpeech 2000 was the only true multilingual SR program, allowing 14 languages to work together

The *SRHM* is also getting matured; previously most of commercial *SRHM*s only support speaker dependent SR technique and isolated words. Now you can find some of the *SRHM*s available in the market, which can support speaker independent SR technique and also the continuous listening. Table 2.4 shows some of the SR hardware modules (*SRHM*s).

For our project we have used SpeechStudio Suite for PC based Voice User Interface (VUI) and Voice $Extreme^{TM}$ Module for stand alone embedded VUI for the Robotics

control.

***SR* programs for developer**	**Vendors**
IBM Via Voice	IBM http://www306.ibm.com/software/voice/ viavoice/
Dragon Naturally Speaking 8 SDK	Nuance http://www.nuance.com/naturallyspeaking /sdk/
Voxit	http://www.voxit.se/ (Swedish)
VOICEBOX: Speech Processing Toolbox for MATLAB	http://www.ee.ic.ac.uk/hp/staff/dmb/voicebox /voicebox.html
Java Speech API[a]	Sun Microsystems, Inc http://java.sun.com/products /java-media/speech/index.jsp
The CMU Sphinx Group Open Source Speech Recognition Engines[b]	http://cmusphinx.sourceforge.net/html/cmusphinx .php
SpeechStudio Suite[c]	SpeechStudio Inc. http://www.speechstudio.com/

[a]JSAPI works with third party SR product from the - *Apple Computer, Inc. ,AT&T, Dragon Systems, Inc. , IBM Corporation , Novell, Inc. , Philips Speech Processing, Texas Instruments Incorporated*. Sun does not ship an implementation of JSAPI

[b]This product is an outcome from Sphinx Group, which has been funded by Defense Advanced Research Projects Agency (DARPA) in the Sphinx projects

[c]Use Microsoft SAPI 5.0 Speech Engines

Table 2.3: Some of the available *SR* programs for developer and their vendors.

SR Module	**Manufacturer**
Voice $Extreme^{TM}$ Module	Sensory,Inc. http://www.sensoryinc.com/
VR $Stamp^{TM}$ module	Sensory,Inc. http://www.sensoryinc.com/
HM2007 - Speech Recognition Chip	HUALON Microelectronic Corp. USA
OKI VRP6679 - Voice Recognition Processor	OKI Semiconductor and OKI Distributors Corporate Headquarters 785 North Mary Avenue, Sunnyvale, CA, 94086 2909
Speech Commander - Verbex Voice Systems	Verbex Voice Systems 1090 King Georges Post Rd., Bldg 107, Edison NJ 08837, USA
Voice Control Systems	Voice Control Systems, Inc. 14140 Midway Rd., Dallas, Tx. 75244, USA http://www.voicecontrol.com/
VCS 2060 Voice Dialer	Voice Control Systems 14140 Midway Rd., Dallas, Tx. 75225, USA http://www.voicecontrol.com/

Table 2.4: Some of the Available SR Hardware Module and their Manufacturer.

2.3 VUI (Voice user interface) in Robotics

User interface is an important component of any product handle by the human user. The concept of robotics is to make an autonomous machine, which can replace human labor. But, to control the robot or to provide guide line for work, human should communicate with the robot and this concept conclude the Roboticist to introduce User Interface to communicate with robot. In the past decades GUI (Graphical User Interface), Keyboard, Keypad, Joystick is the dominating tools for Interaction with machine. Now there are several new technologies are introducing in Human machine interaction filed; from them *SR* system is one of the interesting tool to the researchers for interaction with machine. The reason - it (*SR* system) draw attention to the researcher, because people are used to communicate with Natural Language (NL) in the social context; so this technology can be widely-accepted to the human user fairly and easily. The Roboticist is also getting interest in *SR* system or VUI (Voice User Interface) for the same reason. With the addition of Hearing Sensor (*SR* system), the concept of humanoid robot [5] also becomes true.

After near about three decades of research, *SR* system is getting more mature to use as a User Interface (UI). Scientists are still working to overcome the rest of the problem of *SR* system. Now there are several project going on to introduce *SR* system as a *UI* in Robotics [30, 6, 22, 20, 11, 17]. Most of the projects are working on the Service Robot and focus on the novice user for controlling or instructing the robot. It is easier to introduce to the novice user rather than GUI, Keyboard, Joystick etc. technologies. This is because, human are used to give voice instruction (like - "Go to the Office room and bring the file for Me.") in every day life. But the challenge of HRI is that the novice user only knows how to give instruction to a human; so the research goal is to make the robot capable enough that it can understand the same high-level instruction or command.

For the software development, the normal practice is - to design UI at the early stage of the designing process, then design and develop the software based on the UI design. The concept of UI depends on the robot's sensors in robotics. The spoken interface is very much new component added in the HRI field. In the social context people expect that the robot/machine should understand unconstrained spoken language, so the question of interface requires to be considered prior to robot design [6]. Like - *If a mobile robot needs to understand the command "turn right at blue sign", it will need to be provided with color vision* [6]. Another important thing is that the instructions should be related to the robot's structure or shape, for example - if the robot's structure is a car shape then the instruction should be correspondence to the car driving environment. People have already adapted the scenario of giving instruction from the social context, so when they see the car environment, they normally interact with the car (robot/machine) depend on the environment. *Continuous testing with user is extremely important in the design process for service robot. The instruction design for robot should not focus on only on the individual user, but that other members in the environment can be seen "secondary users" or "bystanders" who tend to relate to the robot actively in various ways* [17]. To know about the environment object is one of the important criteria in robot navigation.

When the user give the instruction like "Go to my office", then it should understand the object "my office"; it is the natural description of an object in social context [30]. From the HRI points of view - the Robot should understand of its environment and its task.

One of the important components of spoken interface is microphone. Microphone hears everything. But most of the noisy data is handled by the SR system. So the designer should careful about the irrelevant instruction for a specific environment, like if the robot stands in front of a wall and it receives the instruction "go ahead", then it should inform the user about the situation.

Another component is Speaker (Loud Speaker). If anything goes wrong then the Robot can inform the user through Speaker (Loud Speaker) using Speech synthesizer (See detail section Speech). For example, if the Robot doesn't understand the command, then it can give the feedback to user through speech or dialogue - like, "I don't understand" using Speech synthesizer.

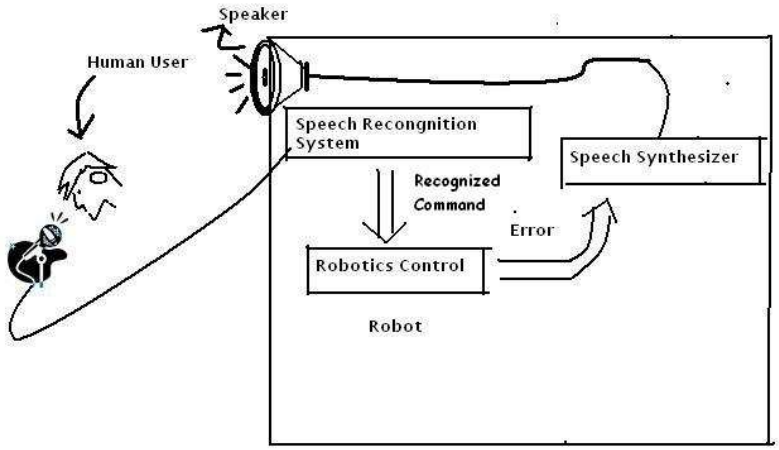

Figure 2.2: Typical Spoken Natural Language Interface in Robotic.

Figure 2.2 shows a general overview of Spoken Natural Lnaguage Interface for Robotics Control.

At the beginning researchers have worked with the *simple grammar sentence instruction*, like "Move", "Go ahead", "Turn left". One of the examples is VERBOT (Isolated word speaker dependent Voice Recognition Robot), a hobbyist robot, sold in the early 1980's - it is not available in the market [13]. Now the researchers have emphasized on *complex grammar sentence instruction*, which people normally use in their daily life [30, 6, 22, 20, 11, 17]. We have also organized our project work in the same way. The roboticists also have used speech synthesizer for error feedback. LED or Color light can also be used for user feedback, but it is not suitable enough for feedback to human user. We have also organized our project work in the same way.

Chapter 3

Language and Speech

A **language** is the system of communication in *speech* and *writing* that is used by people of a particular country or area. [26]

In short we can say language is a systemic way of communication using sound and symbols. From above the definition it is clear that speech is one of the important media of communication, but it should be used in a systemic way - means should follow rules or grammar - then we can say this as a "**language**". So grammar is also an important part of a language.

The way we communicate through speech is called spoken language, more specific - *(language) communication by word of mouth* [37]. In spoken language communication, there are two important things - one is speech and other one is speech understanding. *Something spoken* [37] is called Speech and after hearing if the person understand - what is spoken? - Then it is speech understanding. In the social context we use Natural language as a spoken language. Now the question arrives - what is Natural Language? People are social beings and language is the communicating way between people, we normally call it Natural Language, more specific - *a language that has developed in a natural way and is not designed by humans* [26].

One of challenging research part of Artificial Intelligence (AI) is *Understanding Natural Language*. It is not just a matter of looking up words [24]. The main challenge is to find out the appropriate meaning for the particular situation. So when the question of User Interface (UI) as a spoken language is arise - Understanding Natural Language also an important issue. Other issues are understood the spoken word and speech synthesis. The improvement of SR system makes roboticist interested to choose Spoken Language as UI. Now there are several commercial SR system products are available in the market (See details in Chapter 2: Section 2.2 on page 4). These products have build-in Speech synthesizer. For the proper Speech Recognition (SR) and Natural Language Understanding, these products have used Context free grammar (CFG) (see detail section 3.2). Still there are more improvement is needed in SR and NL understanding area.

3.1 Speech

Speech is an essential component of spoken language. From the early discussion about spoken language, we figure out that Speech Understanding and Speech are two important components of spoken language. In term of machine, the scientist defines these two components as Speech Recognition system and Speech synthesizer. Below we continue our discussion about these two components.

3.1.1 Speech Synthesis

Speech Synthesis is the process of producing sound/speech through the machine [13]. In other words, it makes the machine capable to create speech and we can call this machine *Speech Synthesizer*. It is tremendous aid to give feedback to the user.

The earliest *Speech Synthesizer* was invented by Thomas Edison in 1878. [21] He introduced the record-player or the Phonograph (talking machine), which is one kind of *Speech Synthesizer*. The mechanism of a record-player is to record voice/speech and also possible to playback the voice/speech. Due to advances in technology, now you can even create voice/speech from text. This technique is called *Text-to-Speech Synthesis*, in short TTS.

TTS is computer software that converts text into audible speech [3]. It is a separate technology from speech recognition, TTS is for talking and *SR* is for listening. Both systems have some shared technology; that's why, the manufacturer or developer construct combined products. TTS is available only for the SRSP technology. For the *SR* Hardware Module (SRHM), the Speech Synthesizer normally uses digitized voice recording mechanism. The main advantage of digitized voice recording mechanism is - the sound/voice can be store in the computer's memory. [13]

3.1.2 Speech Recognition System

The process of a machine's listening to speech and identifying the words is called *Speech Recognition System*. We have discussed this technology in detail Chapter 2:Secition 2.2.

3.2 Grammar

One of the key components of a language is Grammar. A Grammar is *the rules in a language for changing the form of words and joining them into sentences* [26]. In another words - *grammar is a body of statements of fact - a 'science'; but a large portion of it may be viewed as consisting of rules for practice, and so as forming an 'art'* [25]. The main point is - it's a way of structuring words to make sentences meaningful.

A SR technique recognizes words, which are spoken. If it is a sentence - then it recognizes the series of word. To identify the meaning of the sentence we need help of the grammar. The grammar helps us to organize the word to make it meaningful. For this reason, the SR system (only in the SRSP) allows the developers to add grammars, which is called language models or artificial grammars. Another reason is, *when speech is produced in a sequence of words, language models or artificial grammars are used to restrict the combination of words* [8]. We can say it another way - *A grammar describes*

a collection of phrases for which the speech recognition engine should be listening.[34]

The simplest artificial grammar can be specified through finite automata and more general artificial grammars (approximate natural language) are specified in terms of a context-sensitive grammar [8]. Most SR systems have used CFG for natural language processing, since CFG have been widely studied and understood and also well efficient parsing mechanisms have been developed for the CFG [23]. The theory of context-free languages has been extensively developed since 1960's [16]. *A CFG is way of describing language by recursive rules called productions* [16]. A CFG (G) is represented by four components $G = (V, T, P, S)$ where V is the set of variables, are called non-terminals, T are called terminals (a finite set of symbol), P the set of productions, and S the start symbol [16].

1. $S \rightarrow I$
2. $S \rightarrow S+ S$
3. $S \rightarrow (S)$

4. $I \rightarrow a$
5. $I \rightarrow b$
6. $I \rightarrow Ia$
7. $I \rightarrow Ib$

Figure 3.1: A context-free grammar for simple expressions (i.e., $a+b$ *or* $ab+ba$ etc.)

The above grammar for expression is stated formally as $G = (\{S, I\}, T, P, S)$, where T is the set of symbols $\{+, a, b\}$ and P is the set of productions show in the figure 3.1. In the Figure 3.1, Rule (1) is the basis rule for expressions. It represents that an expression can be a single identifier. Rule (2) to (3) show the inductive case for expressions. Rule (2) presents that an expression can be produced from two expressions and plus sign is a connecting symbol between them; Rule (3) says that an expression may have parentheses around it. Rule (4) through (7) describe identifiers I. The basis rules are (4) and (5); they represent that a and b are identifiers. The rest of the two rules are the inductive case - if we have an identifier, it can be followed by a or b and result will be another identifier.[16]

A context-free grammar production is characterized as a rewrite rule where a non-terminal element as a left-side is rewritten as multiple symbols on the right [29]. i.e.,

$S \rightarrow S+ S$

But in the case of context-sensitive grammars (CSG), *the productions are restricted to rewrite rules of the form,*

$uXv \rightarrow uYv$

where u and v are context strings of terminals or nonterminals, and X is a non-terminal and Y is a non-empty string . That is, the symbol X may be rewritten as as the string Y in the context u...v . More generally, the right-hand side of a context-sensitive rule must contain at least as many symbols as the left-hand side. [29]

One of the complexity measures of a *SR* is the size of the vocabulary and the complexity of the artificial grammars.The *SR* tools give the opportunity to developers to create grammars for their system context. If you think from the Roboticist's point of view, the grammar should be created in the context of the Robot's environment and the Robot's task related. So, before creating the grammar for the SR engine, the Roboticist needs to study the task definition and the users.

Chapter 4

Implementation

The main goal of our project is to a introduce Spoken Natural Language interface for Robotics control. We also set some requirements, which are mentioned in the Introduction Chapter -

- The Spoken Language interface should be in English Language
- The robot should understand the task from the dialogue
- The system should be speaker independent
- The robot should have some user feedback; such as, if the robot doesn't understand the user commands, it gives the user feedback - "I don't understand"
- The robot should understand the dialogue, which are mentioned in the Table 4.1, 4.2 and 4.3.

Table 4.1, 4.2 and 4.3 show the sentences/dialogues we have chosen to evaluate our system. These sentences/dialogues are arranged in the tables on the basis of grammar complexity and robotic activities.

Robotic Activities	**Sentences**
Move	Move
	Move 10 centimeters
Turn	Turn left
	Turn right
	Turn around
	Turn 30 degrees
Follow-wall	Follow wall
	Follow the wall
Stop	Stop
	Stop here

Table 4.1: Simple Sentences for robotic activities.

Robotic Activities	Sentences
Initiate a location	This is room A
Find-out a location	Go to room A
Back	Back
	Back 10 centimeters
Dance	Dance

Table 4.2: Simple Sentences for some complex robotic activities.

Robotic Activities	Sentences
Move and turn	Move 10 centimeters and then turn left/right/around
Turn and move	Turn left/right/around and then move 10 centimeters

Table 4.3: Complex Sentences for robotic activities.

Note: The underlined words are variables,like Move 10 centimeters- here any number can be used in the sentence.

Table 4.1 shows simple sentences/dialogues for simple limited robotic activities; Table 4.2 shows simple sentences/dialogues for complex robotic activities in a limited scope and Table 4.3 shows complex sentences/dialogues for simple robotic activities in a limited scope.

To achieve our goal, we organize our project in two stages. At Stage I - we studied the related works and also found suitable components (Software and Hardware components-see details in Appendix-A) for the implementation stage. In Stage II - we did the implementation. At implementation, we did the development in two Phases. In the First Phase - we have worked with the SRHM and in the Second Phase - we have worked with the SRSP. In the both Phases we worked with a same Small Mobile Robot named Khepera.

4.1 General Robotic Design

The challenging parts of the prototype development are - implement the Robot's intelligence and make a bridge between the *identified commands through SR tool* and the *Robotic intelligence*. To implement Robotics intelligence we have followed the Hybrid deliberative/reactive paradigm.

Reactive paradigm has got popular in end of 1980's, because of the *faster execution time* characteristic, but still there are limitations caused by eliminating the *Planning*. To overcome the limitation, the Hybrid deliberative/reactive paradigm emerged in the 1990's [24]. *Purely reactive robotic is not appropriate for all robotic application* [2]. The Hybrid paradigm is capable of integrating deliberative reasoning and reactive control

system. This permits the robot to reconfigure the reactive control system based on world knowledge through deliberative reasoning over a world model.

To create a Hybrid paradigm system, we have to identify the behaviors for our robotic control system. For our project we define the behaviors, which are mentioned in the Table 4.4.

Behavior	**Purpose**
Move	Straight robot movement
Turn	For turning
Avoid-Obstacle	Avoid obstacle
Follow-wall	Follow the wall
Move-to-goal	Find-out and follow the goal heading
Obstruction	Identify the obstacle
At-goal	Identify the Goal position

Table 4.4: The behaviors identified for the prototype degin.

These behaviors are reactive behaviors and they are switched according to user commands. If we consider the Table 4.1, 4.2 and 4.3; there we have mentioned Robotic activity wise user's sentences/dialogues. Now we describe the relation between these robotic activities' sentence and the behaviors, which are mentioned above.

If the user gives commands related to *Move* robotic activity, like "Move", the **Move** behavior will be switched on; it makes the robot to forward as default, but the user can also input a distance (centimeter measurement) that makes the robot move this specific distance. For the *Turn* robotic activity's sentences, **Turn** behavior will be switched on. It makes the robot turn and needs the direction, right or left or the number of degrees as input to turn the robot in a specific direction. The **Avoid-Obstacle** behavior helps the robot to avoid the obstacle in its arena. This behavior also toggle with other behaviors, whenever there is an obstacle in front to make the motion safe. The *Follow-wall* activity's command sentences make the robot switch on the **Follow-wall** behavior. This behavior makes the robot following a wall or an obstacle. For the *Initiate a location* activity, the robot stores the current position in the global memory. For the *Find-out a location* activity, **Move-to-goal, At-goal, Obstruction, Follow-wall** behaviors toggle each other depending on the situation. **Move-to-goal** helps to make the robot turn in the goal direction (means the location it's looking for) and to move towards the target direction. The **Obstruction** behavior helps the robot to detect obstruction whenever an obstruction comes in front in the goal direction. This behavior switches on the **Follow-wall** behavior. The **At-goal** behavior helps the robot to identify the goal position and, if positively identified, stop the robot.

After identifying the behaviors, our next move is to organize the behaviors for the Hybrid paradigm. In general the Hybrid architecture has five components or modules - these are [24]:

Sequencer - The agent which generates the set of behaviors to use in order to accomplish a subtask, and determines any sequences and activation conditions.

Resource manager - Allocates resources to behaviors, including selecting from libraries of schemas.

Cartographer - Responsible for creating, storing, and maintaining map or spatial information, and also methods for accessing the data. It often contains a global world model and knowledge representation.

Mission planner - This agent interacts with the human, operationalizes the commands into robot terms, and constructs a mission plan.

Performance monitoring and problem solving - This module allows the robot to notice if it is making progress or not.

We have followed the common components to create the Hybrid architecture for our project. The Table 4.5 below summarizes our Hybrid architecture(Figure 4.1) in terms of the common components and style of emergent behavior:

Hybrid architecture summary (Figure 4.1)	
Sequencer	Reactive planner
Resource manager	Reactive behaviors
Cartographer	Position identifier, Object recognition
Mission planner	Voice User Interface
Performance monitoring and problem solving	Reactive planner
Emergent behavior	Reactive behaviors

Table 4.5: The summary of Hybrid architecture (Figure 4.1) in terms of the common components and style of emergent behavior.

Figure 4.1 presents the Hybrid architecture in our prototype. According to the architecture, *Reactive planner* module works as a *Sequencer* as well as *Performance monitoring* and problem solving agent - this module selects the behaviors from the behaviors-library and sends them to the *Reactive behaviors* module and always monitor the VUI, Position identifier and Object recognition modules inputs to solve the current problem; the Voice User Interface (VUI) module, which acts as a Mission planner, is interacting with the human and send the mission plan to the Reactive Planner; the Position identifier and the Object recognition modules are acting like a Cartographer - the Position identifier always records the current position and the Object recognition module identifies the goal object; the Reactive behaviors acts as a Resource manager. In the Reactive layer, the Avoid-Obstacle module suppresses (marked in the Figure 4.1 with a S) the output from the Reactive behaviors module. The Reactive behaviors module is still executing, but its output doesn't go anywhere; instead the output from Avoid-Obstacle goes to Actuator, when the robot gets obstacle in the front.

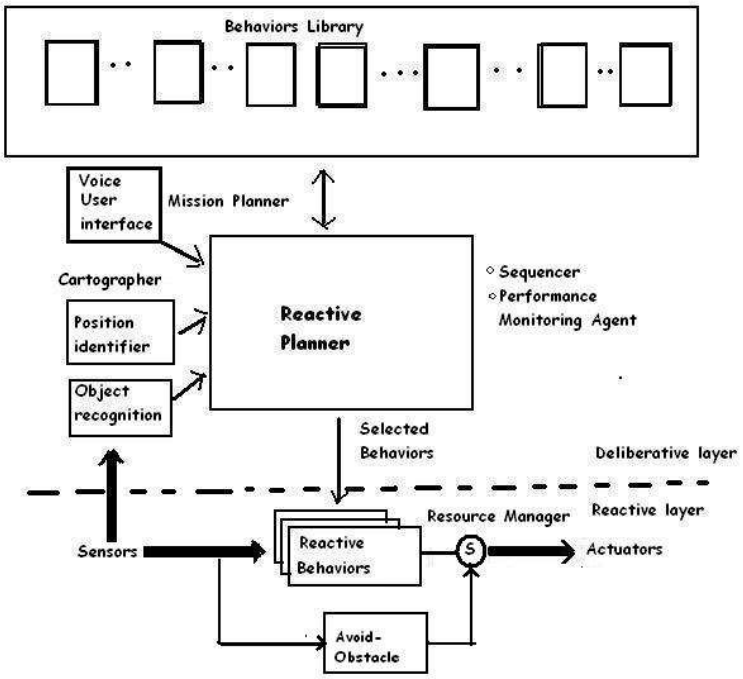

Figure 4.1: Hybrid architecture for our prototype .

4.1.1 Behaviors' Algorithm

We have implemented the behaviors, which are mentioned in the Table 4.4, for both Hardware and Software approach by using the same algorithms. To achieve these behaviors, we have followed different techniques, from which "Breitenberg vehicle" technique [4], Odometry [15] and Bug algorithm [10] are key algorithms We have implemented these behaviors algorithm in terms of Khepera Robot's hardware feature. Here we present these key algorithms below.

"Breitenberg vehicle" technique : The following function have used to implement a "Breitenberg vehicle" for the Khepera [18]-

$$m_L = \sum_{i=1}^{8} w_i \cdot r_i + w_0$$

$$m_R = \sum_{i=1}^{8} v_i \cdot r_i + v_0$$

Here w_i, w_0, v_i, v_0 mean weights, r_i means IR sensors reading and m_L and m_R are the speed for Left and Right Motors of the Khepera. This equation helps us to create *Avoid-obstacle* and *Follow-wall* behaviors.

Odometry: Odometry is used for determine the current khepera position (x-coordiante, y-coordinate, theta). In this algorithm, the *set_position* function is called to set the initial khepera values for x, y and theta. The *read_position* function is used to obtain the tick counts. This tick count values are used to compare the kinematic movement of the left and the right wheels of the khepera. We have followed the below equations to calculate the position from the tick counting [15].

$$R = l/2(n_l + n_r)/(n_r - n_l)$$

$$\omega\delta t = (n_r - n_l)step/l$$

$$ICC = [ICC_x, ICC_y] = [x - Rsin\theta, y + Rcos\theta].$$

$$\begin{pmatrix} x' \\ y' \\ \theta' \end{pmatrix} = \begin{pmatrix} cos(\omega\delta t) & -sin(\omega\delta t) & 0 \\ sin(\omega\delta t) & cos(\omega\delta t) & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x - ICC_x \\ y - ICC_y \\ \theta \end{pmatrix} + \begin{pmatrix} ICC_x \\ ICC_y \\ \omega\delta t \end{pmatrix}$$

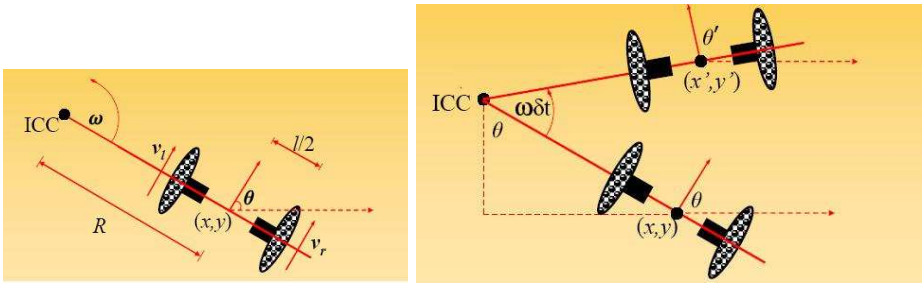

Figure 4.2: Forward kinematics for the Khepera Robot [15]

Where (x,y,θ) is previous robot postion and the new calculated postion is (x', y', θ'). ICC (Instantaneous Center of Curvature), ω angular velocity and δt represent time. Wheel encoders give decoder counts n_r and n_l; *step* is the length (mm) of one decoder tick. (See Figure 4.2)

Bug algorithm: This algorithm is used in making the robot navigate from the source position to the destination position.

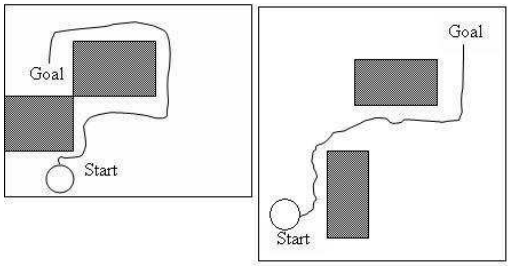

Figure 4.3: The robot can able to handle this kind of situations through Bug algorithm [14].

In the algorithm, there is a while loop that checks if the goal is actually been reached or not. When ever the goal position is not reached the khepera checks for obstacle. If it meets with an obstacle then it follows the obstacle by using *followobstacle* function. If it doesn't encounter an obstacle then it uses the *move2goal* function to move towards the goal direction. The speed of left and right wheel is obtained from either *followobstacle* function or *move2goal* function. Then the *Set_speed* function is called to make the khepera move with the obtained wheel speeds. The current position is updated and the khepera stops when it reaches the goal. [14, 10]

4.2 Hardware Approach

In this approach our main goal is to introduce Speech Recognition Hardware Module (Voice $Extreme^{TM}$ (VE) Module) as VUI for robotics control. Here we have made interface between VE module and General I/O turret; then mounted the turret with three LEDs (Red, Green, Yellow) and a microphone on the head of the Khepera; we have the Robot program in PC and the Khepera is connected through serial cable with PC to receive and send the data for control the robot though **sercom protocol** [19]. The LEDs are used for user feedback. (Figure 4.4 shows a overview of this approach and Figure 4.6 shows the picture of Khepera robot with VE module, LEDs and Microphone)

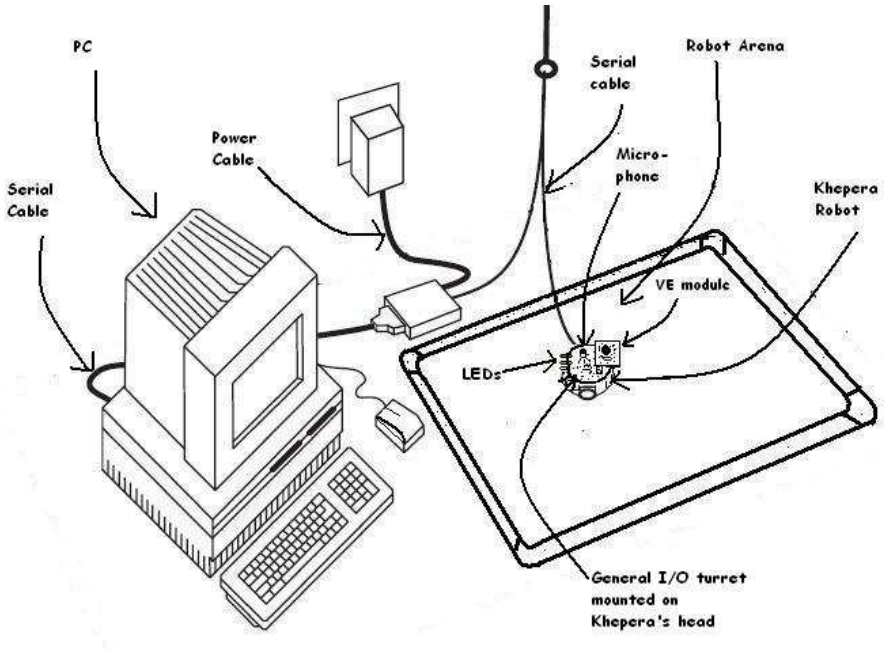

Figure 4.4: Overview of Hardware approach system.

Hardware Components: Khepera (Robot), Voice $Extreme^{TM}$ Toolkit (Voice $Extr-eme^{TM}$ (VE) Module, Voice $Extreme^{TM}$ Development Board with built-in microphone and speaker) Microphone, LED.

Software Components: KT (K-Team) Project, Voice $Extreme^{TM}$ Toolkit (Voice $Extreme^{TM}$ IDE, Quick $Synthesis^{TM}$), MATLAB 7.0.4.

In the beginning we have studied the above mentioned software and hardware components (see details in Appendix A). After that we have designed a work outline for this development phase. We have defined spoken dialogue's simple grammars for SRHM, since it is not capable to load a large vocabulary. The reason behind that is memory space problem. At first the mechanisms of the Khepera and the *VE* Module have been investigated, after that the interface and communication way between the *VE* Module and Khepera has been also investigated.

4.2.1 System Component

Khepera (Robot)

Form the Khepera's Programmer Manual, we found that there are two approaches for programming with the Khepera - one is through sercom protocol, which allows the user to control the robot from any standard computer based on ASCII commands, and other one is through GNU C Cross Compiler, for embedded applications [19]. We have used both of the techniques in this phase. Because ASCII commands can be used from any programming language (we have used MATLAB), which have the serial port communication option and therefore it is easy to use for debugging purpose. Whereas GNU C Cross Compiler is hard for debugging, other then the syntax errors, because developers need to upload the program in the ROM/EPROM of the Khepera (Robot) and then test the functionality of the program.

About the Khepera hardware, it has 8 IR and ambient light sensors, microcontroller and 2 DC brushed servo motors with incremental encoders and wheels [19]. With the help of these IR sensors and others hardware components, we have implemented the behaviors mentioned in the Table 4.4. After studying the General I/O Turret, we have found way of communicating with an external device from the Khepera. Through the General I/O we can only transfer/receive 8 bits (1 byte) of data from the Khepera. (see details in Appendix A)

Voice $Extreme^{TM}$ (VE) Module

Voice $Extreme^{TM}$ (VE) Module is a SR hardware module. The reason we choose this module is that it can support continuous listening and Speaker dependent/independent SR. There are some limitations of this module; the Speaker independent (SI) feature can not be fully controlled by the developer. To introduce SI feature to the VE Module the developer need a WEIGHTS file, which is used to guide the neural-net processing during SI Recognition [32], for every word or phrase. The problem is that SI weights files must be created by Sensory linguists [32]. For our project, we inquired about the Weight files to the Sensory linguists; in response they suggested their new product VR $Stamp^{TM}$ module - where they give the developer freedom to build a SI interface. So we have decided to implement only Speaker Dependent (SD) feature. Also the continuous listening feature is not as good as we expected. VE Module has the 34-pin connector, from these 11 pins as for I/O, as well as connections for a power, microphone, speaker, and logic-level RS232 interface [31]. We have decided to use 7 pins for communication with Khepera and made an interface with a 34-pin header connector with 0.1" centers

to carry signals between General I/O Turret and the VE Module. We have selected P1-0 to P1-6 as output pins; P0-1, P0-3, P04 as a Red, Yellow and Green LED output and P0-7 as a "Training mode" selection pin (it is also set as a input pin) from the 11 I/O pins and pin 4 is for MIC IN (this is a default pin for Microphone input). (See the detailed pins configuration in Appendix - A).

To start writing project application for the VE Module - we have needed to get used to Voice $Extreme^{TM}$ Toolkit. This Toolkit has some hardware components and some software components, which are we mentioned at beginning of this section. Now we will discuss some details about their usage. -

The VE Development Board is an interface for uploading application program to the VE Module and also for training (only for Speaker dependent) and testing the application, which is uploaded. A VE application consists of a program file with any data files - it needs, linked together into a binary file that can be downloaded to a 2Mbyte flash data memory. The developers have to write this application to VE-C, which is a VE language, similar to ANSI-standard C. VE IDE is the development environment for creating VE-C. The VE data files are :

- Speech synthesis files, also known as vocabulary tables (.VES file)
- Speech sentences files (.VEO files)
- Weights files, for use with Speaker Independent recognition (.VEW file)
- Notes and tunes files, for use with the Music technology (.VEM file)

We have used the first two data files for our application. "*.ves" data file was used for speech synthesis technique, it is a speech table. Quick $Synthesis^{TM}$ was used to produce a speech file, "*.ves". "*.veo" data file is used for Sentence generation from one or more speech tables ("*.ves" files). We have used "*.veo" file for speech synthesis in the training session. [32]

4.2.2 System Design

The Figure 4.5 shows the overview of the interface between Khepera General I/O Turret and VE Module. The four areas are marked there. These are -

1. Serial line (S) connector - For interface with the PC.
2. I/O connections area - We only use the Input pins.
3. Free connections area - We have setup LEDs there.
4. Module Connector - Uses for interfacing with other devices

We have intended to use LED to give the developer feedback about the communication status and the device status. Red LED informs the status about CL feature of the SR module, Yellow LED gives the developer status whether the device is "ready" for the listening or not. The Green LED gives the status of Recognition or not. As a consequence of using the SD feature, we have needed a pin for mode selection. In the above we mention it as a "Training mode" selection pin. To use the SD feature we need

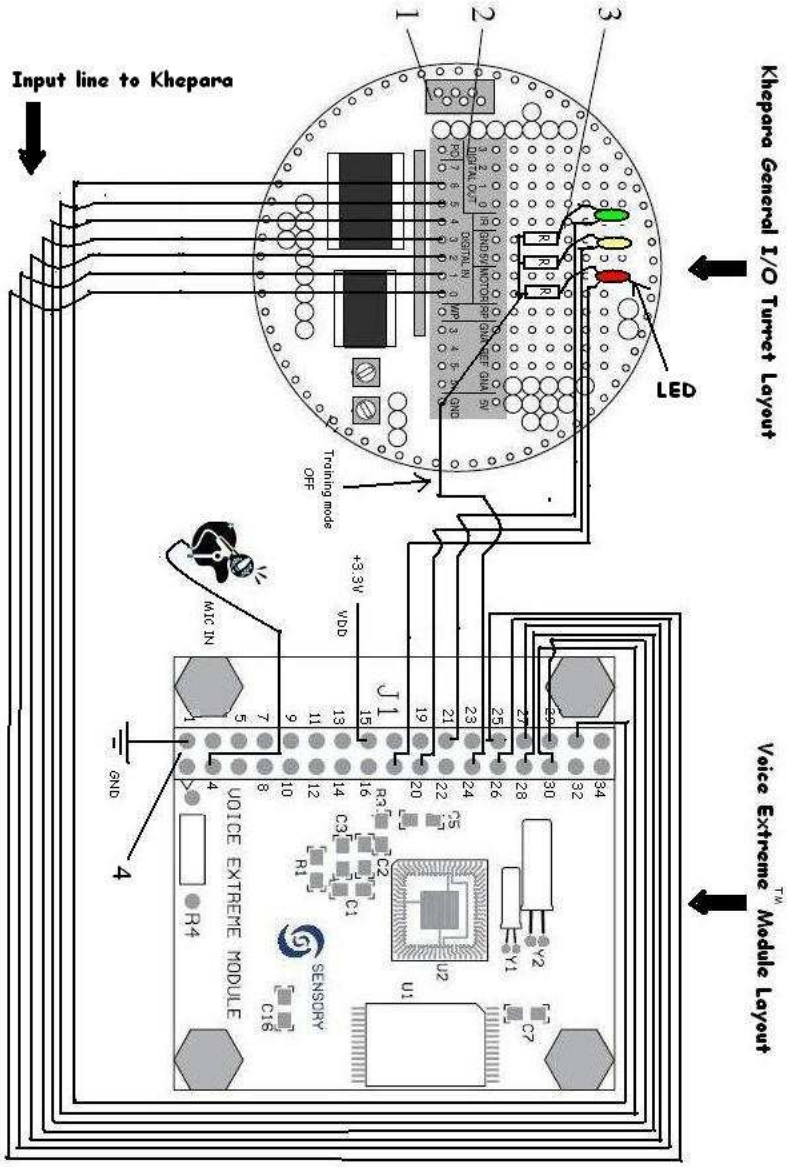

Figure 4.5: The circuit diagram of the interface between Khepera General I/O Turret and VE Module .

a training session to store the voice templates of the user for the every word or phrase. When this pin is HIGH, it set the device for the training session and LOW sets it to the SR mode. Figure 4.6 shows the picture of Khepera with VE Module after implement the circuit design.

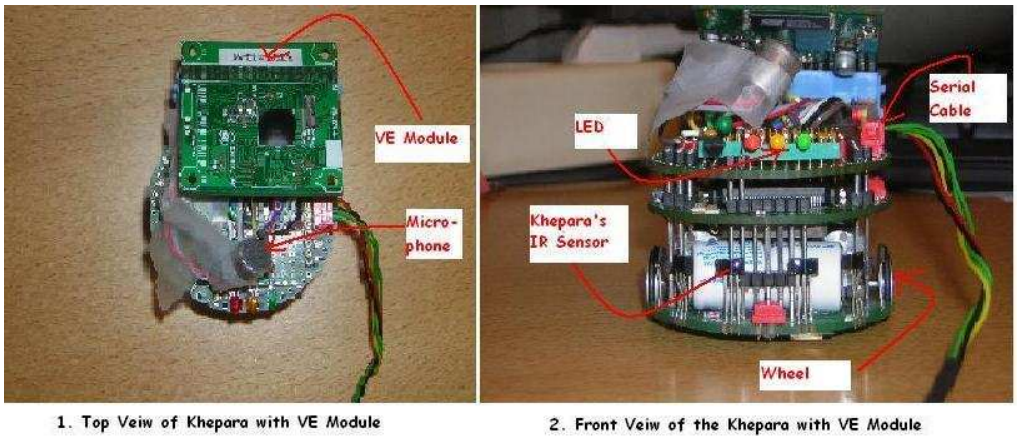

Figure 4.6: The picture of Khepera with VE Module.

Communication Protocol

For data communication between the Khepera and the VE Module, we have chosen packet sending technique. Maximum size of the command-sentence-packet is 6 bytes; starting with a number 127/126 and ending with a number 127/126 - but starting and ending number is the same. Any of these numbers is is selected from these two (127/126) depending on the previous packet's start/end number. i.e., if the previous packet's starting and ending number is 127 then the next newly generated packet's starting and ending number is 126. When the power is switched on, the first recognized (through the VE Module) command-sentence-packet's starting and ending number is 126. (See Figure 4.7)

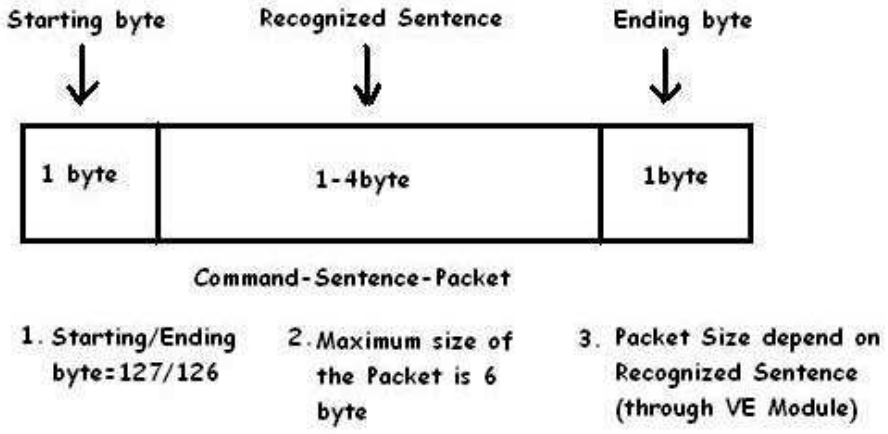

Figure 4.7: Command-Sentence-Packet's Structure.

The starting and ending number help us to identify a packet's starting and ending. The reason we have chosen two different numbers is to identify the last generated packet, because the last generated packet is the new command for the Khepera.

Language Model

Language model/artificial grammar is an important issue for the Speech Recognition system. The problem with SRHM (here, it is the VE Module) is that the developers have to take care of this matter, when they do the design and implementation parts. We have also designed a language model for our system; we have made it for a limited scope - first we have selected some words/phrases, which fulfill our goal, for system and then designed a Lexicon table and the artificial grammars, which are presented below.

Command	**Parameter**			**Unit**	**Object**
	Number	**Identifier**	**Define**		
move (U1)	0	A	clockwise (default 90 degree)	centimeter (U1)	room (O1)
turn (U2) /(O1)	1	B	anti-clockwise (default 90 degree)	degrees (U2)	
go to (U1)	2	C			
stop	3	D			
	4				
	5				
	6				
	7				
	8				
	9				
	10				
	90				
	180				
	360				

Table 4.6: The Lexicon for the language model.

Grammar

1. Command + Parameter (Number) + Unit
2. Command + Parameter (Define default value)
3. Command + Parameter (Define) + Parameter (Number) + Unit
4. Command + Object + Parameter (Identifier)
5. Command

Figure 4.8: The Grammar for the language model.

Semantic Analysis

Check the mapping between Unit/object and Command to find the proper meaning of the sentence and the proper function to run.
i.e From the lexicon we find the mapping like U2=U2, means if "degrees" word come in a sentence there should be "turn" word in the same sentence

Figure 4.9: The Design for Semantic Analysis.

Table 4.6 shows the words/phrases selected for the system design, these are also used at the training session. The User of the system has to train the system following this Lexicon table. There are some marked signs used near the word or phase - like U1, U2, O1; these marks are useful for semantic analysis (see Figure 4.9).

The Figure 4.8 presents the artificial grammars for the SR system. Using these artificial grammars we have done the syntactic analysis at the VE Module, when it's recognized a sentence for system. Example of syntactic and semantic analysis is given below:

"Move 1 centimeter" - this is example a command sentence, which the user can say the robot; the system recognizes the sentence in a sequence of words - "Move", "1" and "centimeter"; after recognizing the sequence of words, the system matches the words' types ("move" - Command, "1" - Parameter, "centimeter" - Unit) in the Lexicon table and sequence the words' type same as the recognition words' order. After that matches the words' type sequence with artificial grammars; i.e., Command + Parameter + Unit. The system also does the semantic analysis; i.e., (move) U1 = (centimeter) U1.

Training Mode

We need to train the VE module, because we are using the Speaker dependent feature. In this feature the User should store his/her voice pattern through a training session. The "Training mode" selection pin activate the training session if it is HIGH, otherwise the system use the previous storage pattern if it is previously trained. We have divided the training session into four steps - in the first step the User has to train the VE Module with "Stop" or similar word command, and then the consecutive steps are trained with the *Command, Parameter, Unit* words. The reason behind these training session steps is - the language model, which we have of this implementation part, consists of *Command, Parameter, Unit* words, like - Move 1 centimeter (*Command+ Parameter+ Unit*) and also the VE Module returns index number of the recognized pattern from the storage table. The training session helps us to identify the index range of the three types of trained words. i.e., 0-5 range indexes are *Command* type words. These ranges are helpful to the Syntactic analysis of the recognized sentence.

4.2.3 Algorithm Description

The algorithms are mainly built on the basis of the components/units, which are used in the system.

Khepera (Robot)

We have followed the general robotic design structure to make the robot intelligent. At first we have implemented the behaviors which are mentioned in Table 4.4. To implement these behaviors, we have followed the "Breitenberg vehicle" technique [4], Odometry [15] and the Bug algorithm [10].

"Breitenberg vehicle" technique [4] helps us to implement the Avoid-obstacle and Follow-wall behaviors. (See more detail in section 4.1.1)

The Odometry gives the Khepera position (x,y,θ)- x,y coordinate, θ is the heading of the Khepera and the Bug algorithm [10] helps to move-to the goal position.(See more details in section 4.1.1)

After building the behaviors which are mentioned in Table 4.4, we have managed the behaviors by following the Hybrid architecture show in Figure 4.1. According to the architecture, the program select behaviors based on the recognized voice command through SR and activate the behaviors. For avoiding collision, we have implemented mechanism that the *Avoid-obstacle* behavior is switched on whenever an obstacle is nearby.

In the Khepera function/module we also read the Command-Sentence-Packet, which is sent by the VE Module. A loop is always checking - is there any new Command-Sentence-Packet generated or not, by checking the numbers 127 and 126 appearance. If at the first time (after the power switched on of the system) 126 appears, the next new generated packet start with 127 and then vice-versa. In the Packet reading we have checked the starting and ending of the packet by check the same number (it should be 127/126 [1]) appears after 1 or maximum 4 (four) different numbers (these number should be with-in 0-125), these numbers represent the command-sentence indexes.

We have the Lexicon-table (see Table 4.6) of words in the Khepera function/module, which is identical with the stored voice-pattern for words in the VE Module. Here identical means that if an index represents a voice-pattern for a word in the VE Module, the same index represents the same word in the Lexicon-table - that means the index numbers, which we read-out from Packet, represent the same words from the Lexicon-table. After identified the words, we have done the semantic analysis to verify the sentence meaning, which means the identified command sentence can be "Move A cm" - here the sentence follows the grammar perfectly, i.e., *Command+Parameter+Unit*; butA is not the correct parameter for the Move command, it should a number type parameter, i.e., 10. If the sentence is meaningful then send the command to activate the related behaviors.

Voice $Extreme^{TM}$ (VE) Module

In this Module we divided the main function in two modules - one is training mode and other one is recognition mode.

[1] The VE Module's 7 I/O pins are connecting to the Khepera for sending data. Through 7 I/O pins we are able to generate any number with-in 0-127. We have reserved 127 and 126 numbers only for the Packet start/end byte, other then these we have used for representing indexes of the words, which are stored in the VE Module.

First we check the "Training mode" pin is HIGH or LOW. If it is HIGH we call the training function.In the training mode, we save the voice-pattern of the user in the Flash memory of the VE Module. At the beginning of the training session we allocate the memory for the voice-patterns, which are to save. There are four steps in the training session. At the first, the first word of the training session should be "Stop" or similar word and it is automatically switch on the next step. We suggest the user to use "Stop" or similar word; because according to our design the user can use this word for finishing the other consecutive steps and also can use as a command word for stop the robot movement. In the next consecutive steps user have the option to train maximum 20 words in every step. At 2^{nd} step user can train the system with *Command* word; according to our Lexicon-table 4.6 he can only able to train 4 *Command* words; so after trained these four *Command* words, he/she can proceed to the next step just simply saying the first step's recorded word - i.e., "Stop". For the voice-pattern sample collecting, we first collect a pattern sample of a word from user by requesting him/her through speech synthesis - i.e., "Say word one"; after collected the first sample, we request again to give another sample by using the speech synthesis - i.e., "Repeat". Then we check the similarity of the two samples, if these samples match each other then we take an average of the two samples; otherwise ask for a another sample through "Repeat" request. In the 3rd step user can train the module with *Parameter* words and then the last step the user can train the module with *Unit* words and *Object* words.

After collecting the lexicon through training session, the VE Module is read for Speech Recognition. After collecting the lexicon through training session, the VE Module is read for Speech Recognition. We have applied the *Continuous Listening* (CL) feature for SR. To implement the CL feature, we have used built-in function to recognize a word pattern from the lexicon and to return the index number of the word from the table. We set this built-in to listen 2 second duration and then time out, if it listen a word with-in this duration it waits for another word and so on as far the words sequence follows the Grammar (See Figure 4.8); if the module waits for a word it blinks the YELLOW LED. When the function listens the words it does two things, recognize the pattern and check the grammar; if any recognition or grammar error finds that processing time, it on the RED LED and if everything goes fine it gives the green signal through GREEN LED. After recognition a sentence, it makes a Command-Sentence-Packet by using the protocol (See Figure 4.7) and then transmits the packet after every 2 sec through the output pins as far as the new packet is generated.

4.3 Software Approach

Here we have implemented a VUI for the robotics control through Speech Recognition Software Program (SpeechStudio). In this approach, we have the Robotic Control and Speech Recognition program in the PC; a Microphone is connected to the PC and the Khepera (Robot) is connected to the PC through serial cable. Here we have also used **sercom protocol** [19] to control the Khepera. We have discussed this approach more details below. The Figure 4.10 shows a overview of this approach.

Hardware Component: Khepera (Robot), Microphone, Loud Speaker.

Software Component: Visual Basic 6.0 (VB6), SpeechStudio Developer Bundle (SpeechStudio, SpeechRunner, Lexicon Builder, Lexicon Lite, SpeechPlayer, Profile Manager)

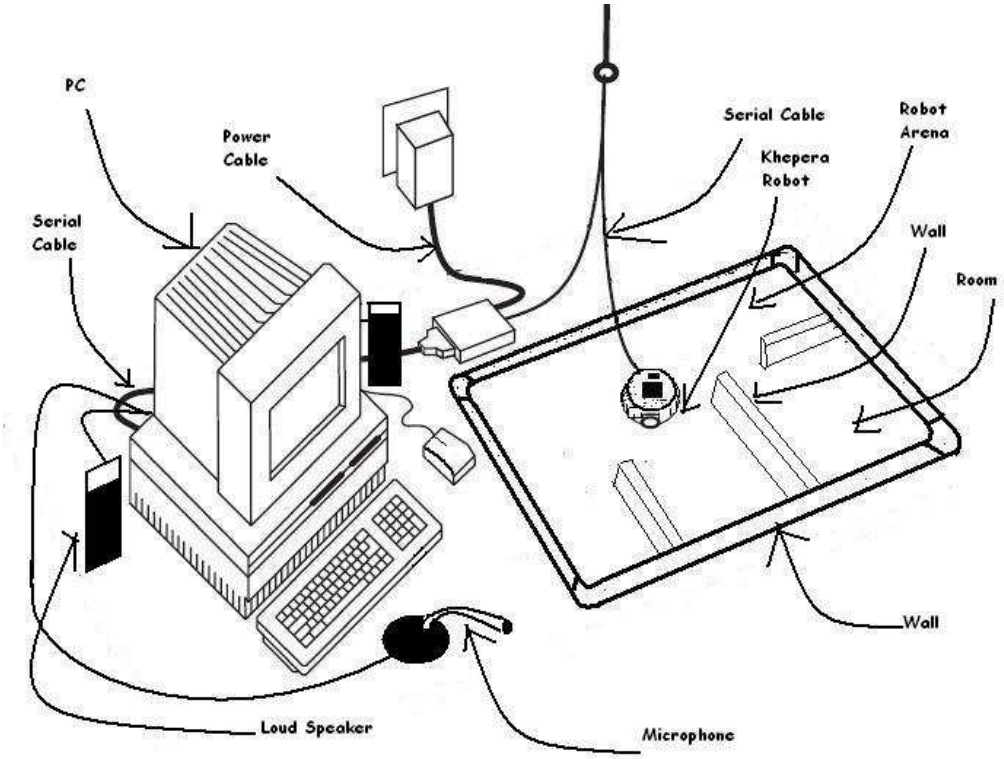

Figure 4.10: Overview of Software approach system.

There are several SR software products available in the market and also these are used commercial with many products' user interface. These SRSPs are more mature then the SRHM and also support large vocabulary and complex grammar. These SRSPs are more mature then the SRHM and also support large vocabulary and complex grammar. That is why; we have chosen to implement another prototype by using SRSP. In this implementation phase our first approach to know about the chosen components. We chose SpeechStudio Developer Bundle as a SR interface, because it is suitable with Microsoft Speech API and our development environment was in Microsoft Windows.

We have done this implementation in two steps. One has been tested with Simple Sentences - i.e., we have presented as a Candy Robot in the Stockholm International Fair and another has been tested with more complex sentences for controlling Robot. (See details the chapter 5)

4.3.1 System Component

For this phase, we have chosen system components that are suitable for SRSP - SpeechStudio as a SR system, the same small mobile robot (Khepera) is also used here, a microphone and a loud speaker.

Khepera

In section 4.2.1 we have mentioned two approaches for programming the Khepera. One is through sercom protocol; other one is through GNU C Cross Compiler [19]. In previous phase (at Hardware approach) we have used both, but for this phase we have only used sercom protocol, which allows the user to control the robot from any standard computer based on ASCII commands [19] and VB6.0 to communicate with Khepera through sercom protocol.

We have implemented the behaviors by following the same strategy mention in section 4.2. The difference is that we implement all behavior using VB6.0 and sercom protocol.

Here we haven't needed to use the General I/O Turret, because we have no external hardware device to interface with the Khepera.

SpeechStudio

SpeechStudio Developer Bundle has six components (these are mentioned above) for the developers to handle. From these, [34] -

- SpeechStudio is used for creating grammar;
- SpeechPlayer is a mediator component between the speech recognition engine and the microphone, it checks the grammar and voice pattern;
- For debugging the SR system SpeechRunner is used;
- Profile Manager is used for adjusting the microphone and creating user profile, this SR system is normal respond with any user - means speaker dependent, but because of noise factor sometimes it needs to be training by the user to adjust with the environment, that is why user profile is important;
- Lexicon Builder is to add new word in the SR system's dictionary and the Lexicon Lite is used to backup the dictionary.

Figure 4.11 shows the interfacing between SpeechStudio SR system and VB6.0. SpeechStudio Suite is an environment for the development of voice user interfaces (VUI) in Microsoft Visual Basic . SpeechStudio Suite has an authoring component called "SpeechStudio", which helps the developer design grammars to describe conversations, and to connect these grammars to actions in his/her programs. The resulting grammar data is involved at runtime via instances of the SpeechStudio Control, which communicate as clients of the SpeechPlayer runtime system. The SpeechRunner is the SpeechStudio Suite's powerful debugger and testing tools.

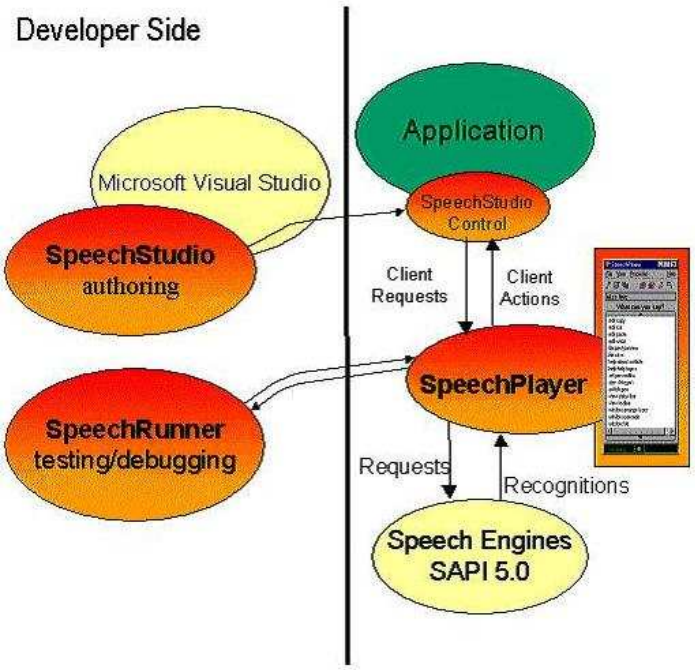

Figure 4.11: An overview picture of interfacing SpeechStudio SR system with VB6.0 [35].

4.3.2 System Design

At the software approach the main interesting design area is the interfacing between SR system and the Robotic application. We have planned to use “Option button” to activate a behavior and a “Text Box” to give the parameters for the activated behaviors; the reason we have chosen ”Option button” and ”Text Box” is these tools can easily handle from the SpeechStudio’s grammar creation feature.

Figure 4.12: An example of “Option Button” and “Text Box” use for “Move” and “Turn” behaviors.

Figure 4.12 and 4.13 give examples of how to control behaviors by “Option Button” and “Text Box” through SpeechStudio (the SR system). Figure 4.13 shows a portion of the grammar file named “Task.gram”, which is written to control the system through speech. This Figure also shows an example, how the developer can create pattern of grammar to control the system components; this pattern specifies that when the application system (*Speech_Khepera*), which controls and communicates with the robot, has the attention

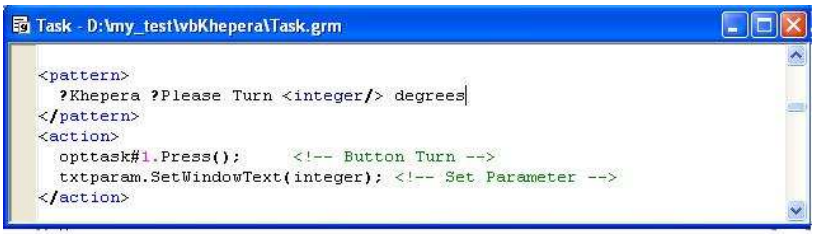

Figure 4.13: An example of create grammar to activate "Option Button" and to send parameter at "Text Box" for "Turn" behavior.

of SpeechPlayer, our system user can say "Khepera Please Turn 30 degrees"; recognition of this phrase will choose the option button - "Turn" named *opttask(1)* (shows on left-side in Figure 4.12) and 30 will be set in the "Text Box" named *txtparam* (shows on right-side in Figure 4.12). To activate the "Turn" -option button in Figure 4.12 we have used *Press()* function and also send integer parameter to the "Text Box" by simply using *SetWindowText(integer)* function within the pattern $< action > \ldots < /action >$; both functions are built-in function of the SpeechStidio program. *The grammar file is an XML file. XML is a general language for exchanging information. Each piece of XML is bracketed by a start token, such as* $< pattern >$*, and a matching end token - in this case* $< /pattern >$*. Empty pieces can be abbreviated to* $< myToken/ >$ *instead of* $< myToken >< /myToken >$[35].

In the example of Figure 4.13 (the "Task.gram"), the grammar pattern has two parts - *Phrase* part and *Action* part. The *Phrase* part starts with the $< pattern >$, which is the start token; then the end with $< /pattern >$ - end token. The phrase, which can be spoken to control the system, is written within $< pattern > \ldots < /pattern >$. In our example, the phrase is - **?Khepera ?Please Turn** $< integer/ >$ **degree**. Here $< integer/ >$ means it can be any whole number - i.e., the user can say "Turn 60 degrees" and ? sign before the word means the word is optional - it can be said with the other words in the phrase, not necessary; but other words should be said to do the action for which the grammar pattern is written - i.e., for the example of Figure 4.13, this grammar pattern is written to activate the Turn behavior option with the degrees' parameter (like - 60 degrees); so the user can say "Turn 80 degrees" or "Please Turn 80 degrees" or "Khepera Please Turn 80 degrees". The Action part is start with the $< action >$ - start token; then end with $< /action >$ - end token; the action, which will be taken to choose the "Turn" - option button (*opttask(1)*) and to set integer type variable to the "Text Box" (*txtparam*) after spoken the phrase, is written within $< action > \ldots < /action >$. The first line - "**opttask#1.Press();**" means that after recognition of the phrase, which is written in the Phrase part; SR system will choose the opttask(1) ("Turn" - option button) and then go the second line - "**txtparam.SetWindowText(integer);**", means set the integer type variable (whole number), which is recognized by the SR system from the phrase.

4.3.3 Algorithm Description

The whole system can be divided in to two parts - SR system part and Robotic application part. SpeechPlayer handles the recognition part based on the grammars,

which we have created SpeechStudio component and send the recognized sentence to the Robotics application part. For coding simplicity we divide the Robotic application program mainly in two modules. Within these modules we have further division in more modules (functions).

From these two modules, one of the module's tasks is to activate the components, make them read to communicate with each-other, switch the behaviors whenever the system needs and also take care about the user interface; we have named it "frmcom". In the other module, we have written the general functions and behaviors function for Khepera, we have named it "Khepcom"; these functions can be called from the other modules of the system. The algorithm describes below on the basics of the two major modules of the system:

frmcom **module**: At starting of the module, we activate the Serial communication to communicate with Khepera through COMPORT#1, and then activate the SpeechStudio components for Speech Recognition. After activation the Khepera and SpeechStudio, we have set the robot position with (0, 0, 0) -means x,y coordinate is set to zero and the heading angle is also set to zero degree, through Odometry function and also give the welcome message to the user. At the same time an activity monitoring module is activated; after activation of this module, it checks the input data every 5 milliseconds. Here the input data means data from Khepera Robot and SpeechPlayer (a component of SpeechStudio). Based on the input data this module calls the behaviors and communicate with Khepera through the function, which are written in the Khepcom module.

Khepcom **module**: In this module we have written the function to communicate with Khepera. Here we have implemented the behaviors through "Breitenberg vehicle" technique [4], Odometry [15] and Bug algorithm [10]; the same way, which we have mentioned in section 4.1.1 to build the behaviors.

Also the **sercom protocol** [19] for communication with Khepera has implemented in this module through different small functions. Such as *F_Khepcom* - for sending and receiving data from Khepera through serial cable, *Set_speed* - for set the Khepera wheels speeds, *KStop*- for stop the Khepera movement, *Read_prox* - Read proximity sensors data. The system has some global memory and some searching function, which are also implemented here - *find_obj* - it is a search function and it find-out the object position, which previously store in the global memory through object identification command, like "This is room A"; global memory like - Khepera Previous position.

Chapter 5

Evaluation

We have to go through a testing phase to discuss the implementation success. In this chapter we are going to present our test plan and the over-all result of the testing phase. The test plan is mainly divided into two parts : the SR interface Hardware approach and the Software approach. We also got an opportunity to test our system in a technical fair. We presented our system as a Candy Picker Robot (we have named it CARO - the Candy Robot) to attract visitors at the fair. There, we also did some usability test. We have separated this section mainly in two parts - one is Test plan and another is Results. These are elaborated below-

5.1 Test Plan

A test-plan is an important part of a testing session. It gives us an outline for testing and evaluating the system. We have designed a test-plan for our system testing and according to this test-plan have done our testing on the system. We have applied two approaches of testing to the system.In the first approach we have tested the system with simple sentence and within simple limited robotics activities, which are mentioned in Table 4.1. We applied this approach to both the Hardware and the Software SR interface for controlling Robot.

In our second approach we have used both complex and simple sentences and also with some complex robotics activities, but in limited scope (See Table 4.2 and 4.3). To design the test-plan, we have considered the implemented grammars and behaviors, which we did in the implementation stage.

To present our system at the fair, we have used the simple activities from the Table 4.1 and also one activity from Table 4.2- the Back behavior; we just limited the sentence making scope with these robotic activities but we have not limited the sentences. We do this to give the user flexibility. i.e., User can make any sentences, like - Robot, please move or Go forward, without using the sentences mentioned in Table 4.1 , 4.2. To achieve this goal, it was one of our duties in the fair to observe the users and track record the users' sentences - whenever a new sentence is used, introduce the sentence to the system afterward. We have also performed usability test of the system in fair. To perform this usability test we have made a user questionnaire (see in the Appendix- C).

5.2 Results

Here we mainly discuss elaborately about our testing experience and the test-results. We have executed the testing phase according to our test-plan, so we have followed the same sequence to present the results and the experiences.

5.2.1 Hardware approach

According to the test-plan we tried to execute the command-sentences, which are mentioned in the Table 4.1. We have only implemented the speaker dependent feature; so before do the testing, we have to go through the training session using the Lexicon-table (Table 4.6), which is mentioned in the Chapter 4. After the training session, we have tested the system with command-sentences (See Table 4.1).

The results of the test are not so impressive. The VE Module's (SR module) speaker dependent feature is very sensitive. For example - if you train the module from a particular distance (distance between Microphone and the User), to get better SR result - in our case to control the robot activities, you have to maintain the same distance with microphone and also the same tone. Otherwise it does not recognize the command-sentences properly. We have also found that the sentences with three words are not always recognized and the LEDs are not a suitable interface to give user feedback.

5.2.2 Software approach

Here we present the test-results of our SR software approach as an interface for robotic control. We have tried to execute all the sentences, which are mentioned in the Table 4.1, 4.2 and 4.3. Through the testing we have found that it is more impressive then the hardware approach result, but we have to keep the noise in minimal level. Another observation is that when we are not planning to communicate with the system, we have to mute or switch off the microphone connection; we have to keep in mind that microphone hears everything, so the surrounding noise can make the system malfunctioning. We have introduced Avoid-obstacle behavior to the robot to protect itself from this type of malfunctioning. Sometimes the system can not response with the user's speech, the reason behind that is mainly the noise factor or the user's speak is not so clear or the user says sometime which is not designed to respond to.

5.2.3 Experience from the Technical Fair

It is a great experience to present the system in the Stockholm International fair'2005 (Tekniska mässan 2005). This fair was open for general people that gave us a huge opportunity to test our system in the public place and also learn people's opinion about the system and VUI for robotic control. It was also helpful to find-out our system problems and limitations.

We presented our system as a Candy picker Robot - CARO. The idea behind that was to give pleasure to the user and make them to use the SR interface for robotic control and gain a candy - like a fun game. We have set a plow in the front of the Robot, by which the Robot can push a candy on a plain surface and also made a cage using plastic glass. That why, if we put the robot and candies inside the cage, then user can see it from the outside and this cage has also a little door in the front, from which

a candy can come out easily. The task of the user is to navigate the Robot to bring a candy for him/her through this little door.

From day one of the fair, the visitors gave as much response as we expected. The people were curious about the CARO and also interested to try for a candy. To know the users' impression and also to do the usability evaluation using the real-time users, we have prepared a user questionnaire. We have also got a lot of response to fill-up the questionnaires from the user.

Figure 5.1, 5.2 and 5.3 shows the CARO's picture from the technical fair. These pictures give you the overview of the CARO's arena.

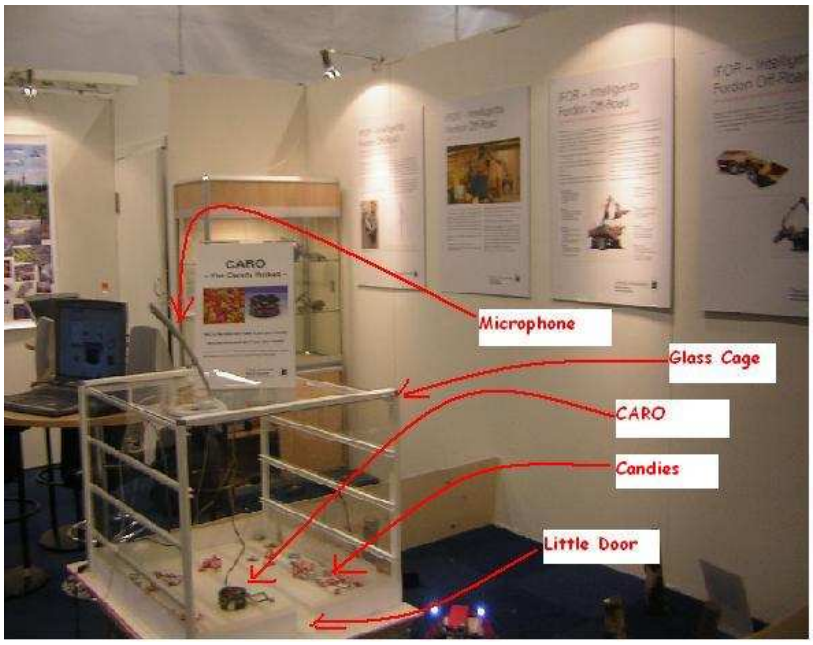

Figure 5.1: The picture of the CARO's arena (outside view)

Usability evaluation

For the usability evaluation of the SR interface for robotic control, first we have identified the usability factors by which we can evaluate the usability of this system. Our chosen factors are:

Learnability - It's a most important factor for any system. We can define the learn ability, how easy it is to learn the system. For our project - how is easy to learn to control the robot through speech. To know the learn-ability factor we have asked the user three following questions:

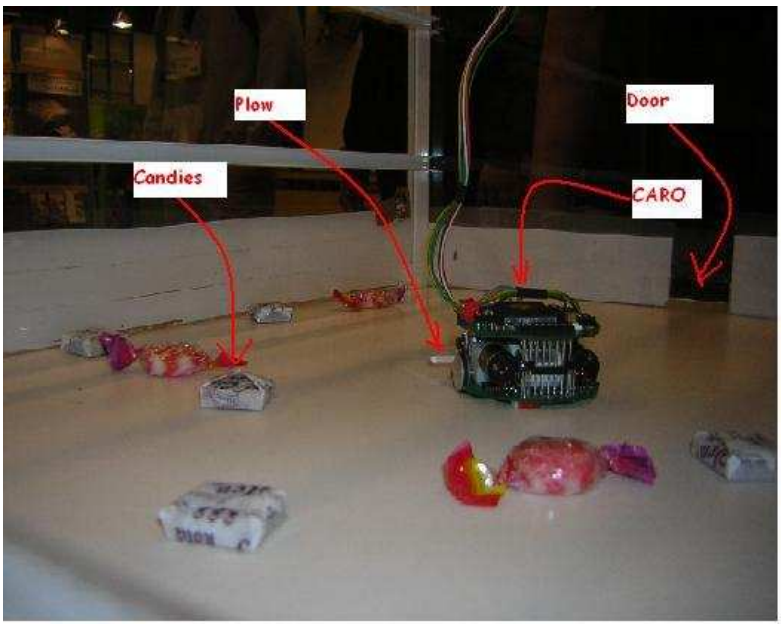

Figure 5.2: The picture of the CARO's arena (inside view)

Figure 5.3: Curious visitors are watching the CARO (The picture from the Technical fair)

- Did you manage to get a candy out?
- If yes, how long time did it take?
- Did you find it hard to control CARO?

Efficiency - If the system gives output that is accepted by the user then we can say that the system works in efficient way. In this case, is the system responding perfectly of the user's speech. To investigate the system's efficiency factor we have asked the user following questions:

- Do you find the delay time disturbing?
- When you told CARO to do something - did it act like you have expected?
- If CARO did not do what you told it, what happened?

Flexibility - We can define flexibility as how well the system enables users to do more things. Our investigation point is to know - are the commands flexible for the users to navigate the robot. To know the flexibility factor:

- Are the commands flexible enough to operate CARO?

User satisfaction - The main goal of any system is to satisfy the user. If the user can do all the things he/she wants from a system that means it's satisfying the user perfectly. It is hard to know the user satisfaction through some specific questions. To investigate this factor we have considered the whole questionnaire (see in the Appendix- C) answers but we have given more emphasis to these following questions:

- How do feel to talk with CARO?
- When you told CARO to do something - did it act like you have expected?
- Would you prefer to control the robot with speech instead of joystick or keyboard?

Before discussing the questionnaire result we present some information about the users, who have participated to test the CARO and also fill-up the user questionnaire; because, the user's information is an important factor in the usability test. But the conclusion we have made from these users information and questionnaires may not reflect all the people in the society; it only reflects the participants at the fair and we also don't know about what types of people's participation was majority at this technical fair. We have analyzed the user through Age, Sex and Occupation; and all this information we have got from the questionnaire sheet. The user's information is presented in histogram in Figure 5.4 and 5.5.

Figure 5.4 shows that young males were most interested to participate in the test. Of

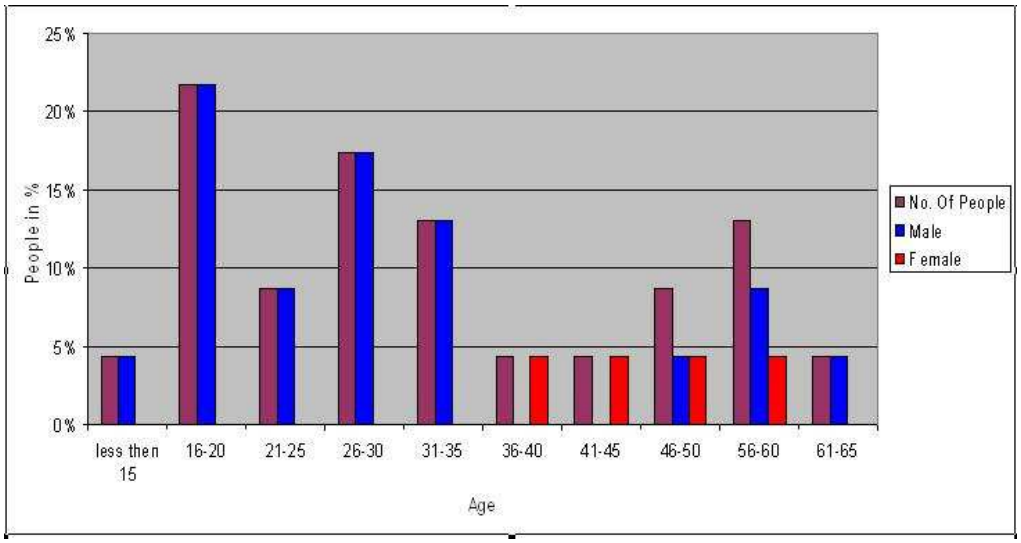

Figure 5.4: The histogram shows the user's information on the basis of age and sex.

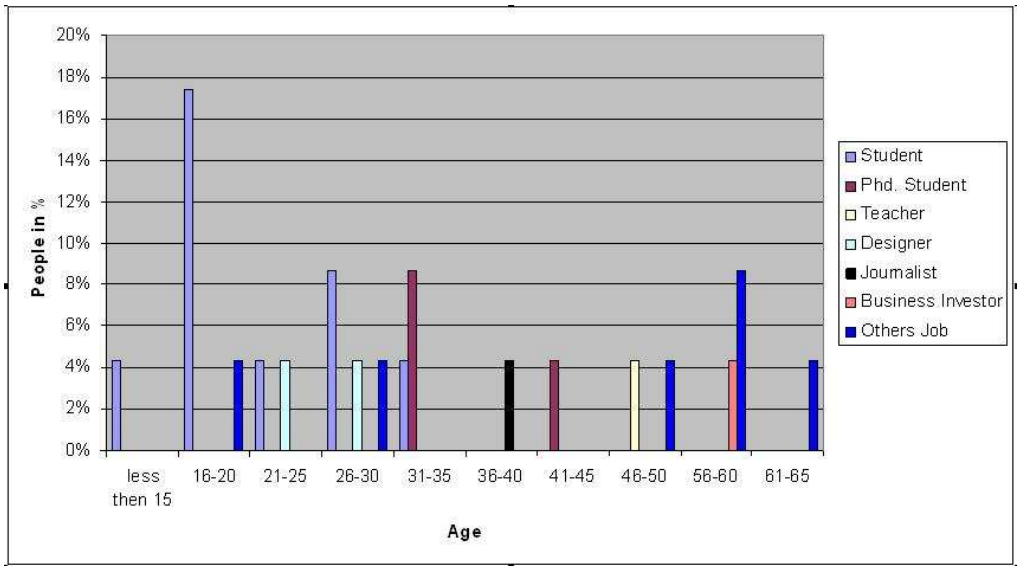

Figure 5.5: The histogram shows participant user's information on the basis of age and occupation.

the females, the aged persons (all above 35 years) have participated. According to Figure 5.5 the most of the participant users were Student and PhD. students. From these two histograms we have also found the different kind of people participation for our system testing. Our project goal is to make a user interface for a Service Robot, which will work in the social context; and also the interface should be for the novice user. This

usability test data is helpful for us, because of the participation of different kinds of people (especially novice user).

To evaluate the learnability factor, we have investigated question 2, 3 and 4 (See in Appendix- C) from the answered questionnaire sheet. From investigation we have found that 65% users have failed to manage a candy out, but rests of the users have got success; the succeeded user have took on an average 5 minutes to get a candy-out. Another interesting thing is that more than 50% of the users have found the task easy. The Figure 5.6 gives the overview of the users comment about easiness and hardness to control the system. The pie diagram shows the overall comments and the histogram show the age-wise comments. From the histogram we have found that almost every age group find the system easy to control. So we can say that the system is easy enough in terms of learn-ability factor.

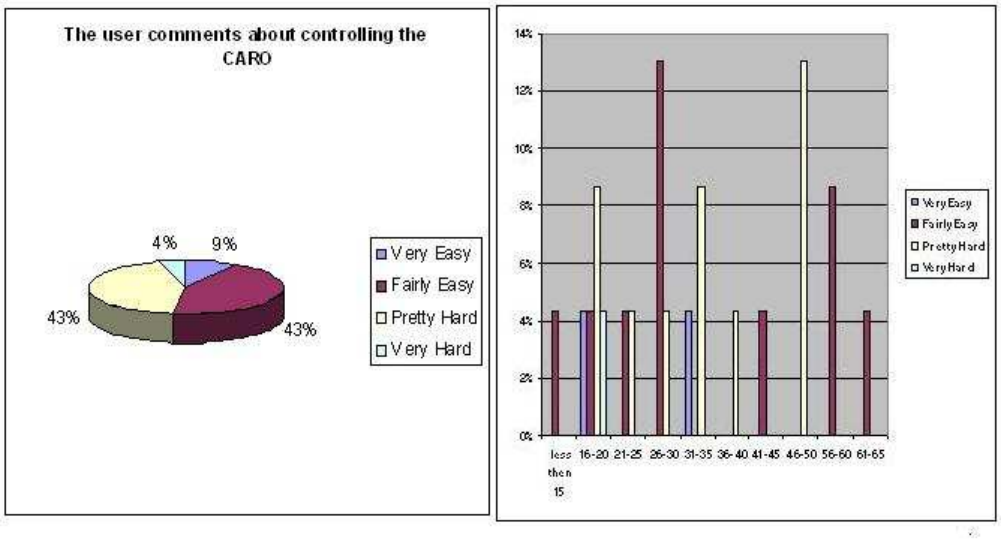

Figure 5.6: The user comments about controlling the CARO.

The system efficiency evaluation is an important factor in the usability test. It gives us the information about problems and limitation of the system. To investigate the efficiency our main focus point is - "Is the Robot perfectly responding the user's speech?", depending on this we have asked Question No. 5, 7 and 8 (See in the Appendix- C) to the users. The answers are showed as pie diagrams a, b and c in Figure 5.7. According the diagram - (a), we have found that after giving a command to the robot, the delay time is not seen as a problem for the users. Only 17% of the user have found that - it takes long time to understand the commands; the majority of the users have felt that - it's not a big problem and the rest of the users have found - it's ok for them. The second diagram - (b) shows that 61% users have found - CARO's responds Often to the command, 22% of the users have found - Seldom and the rest of the users have found - Always. The third diagram shows - what CARO does when it doesn't understand the command. Most of the users say - it does nothings, 52% say - it does something else and

only 4% says - it does right thing, but not perfectly. From these diagrams we can say that - CARO understand the commands often and when it understands - it does the act perfectly. So our finding is that because of SR system recognition problem the system acts in this nature; from the SR documentation [34] we have found that the noise factor effects SR system performance. A Fair is gathering of people, so the noise factor effect makes the system response - Often, not Always.

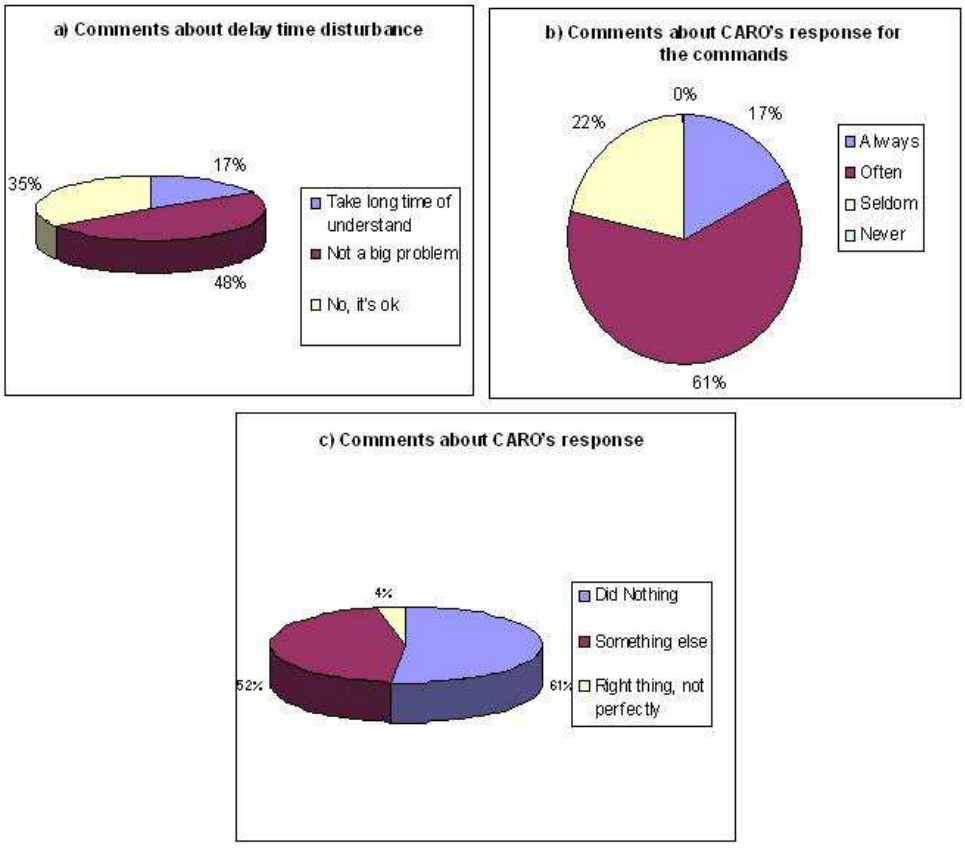

Figure 5.7: The Users comment about CARO's efficiency.

Another usability factor is to know the flexibility of the system from the users' point of view. We have evaluated the flexibility of our system by asking Question No. 6 (See in the Appendix- C) to the user. Our main focus is to find out that the commands are flexible enough or not to navigate the CARO in its arena, are the commands are sufficient or do we need to add more commands. The Figure 5.8 presents the result of the question and shows that 61% of the users believe that the commands are sufficient to control the CARO in its arena, 13% users say don't know, 17% of the users believe that the commands, which already exist, are not sufficient - need to add more, like "Fetch the Candy" and 9% of the users say that they need training to control the CARO. From the result we can conclude that the commands are flexible enough to control the CARO.

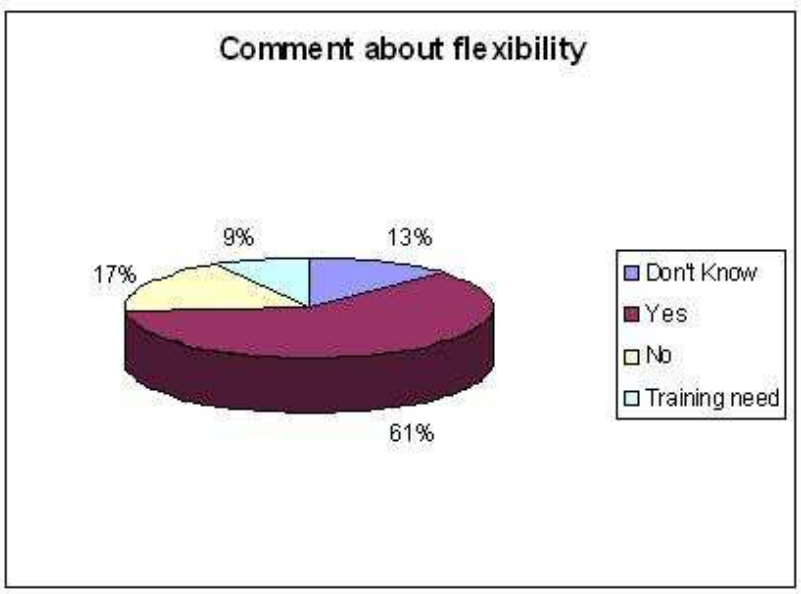

Figure 5.8: The Users Comment about flexibility.

The most important usability factor and also hard to justify from the user answer is User satisfaction. To investigate this factor we have judged all questions' answer. But we have given more emphasize to Question 1, 7, 8, 9 (See in the Appendix- C). We have already discussed the answers of Question No. 7 and 8, when we have investigated the efficiency. Now we are going to discuss the answers of Question 1 and 9. Question 1 is mainly to find out about the feeling when talking with CARO. Figure 5.9 presents the results in pie diagrams. From the Figure 5.9 (a), we have found that 43% finds it fun to talk to the system, 22% feel - it is unusual, 17% of users have found it - Funny, 9% say that it is "Ok" and other users comments that sometime the CARO doesn't recognize the command, they need training to control the CARO and it is hard to know what to say. We have also found the preferences of the users to control the robot in Figure 5.9 (b). It shows that 70% of the users like to use "speech" to control robot, 22% prefer Joystick/Keyboard, 9% say - it depends on situation and 4% say, they don't know. After evaluating all the questions' answers, we have found that majority of the users have given positive answers about CARO, so we can conclude that our system satisfied our users.

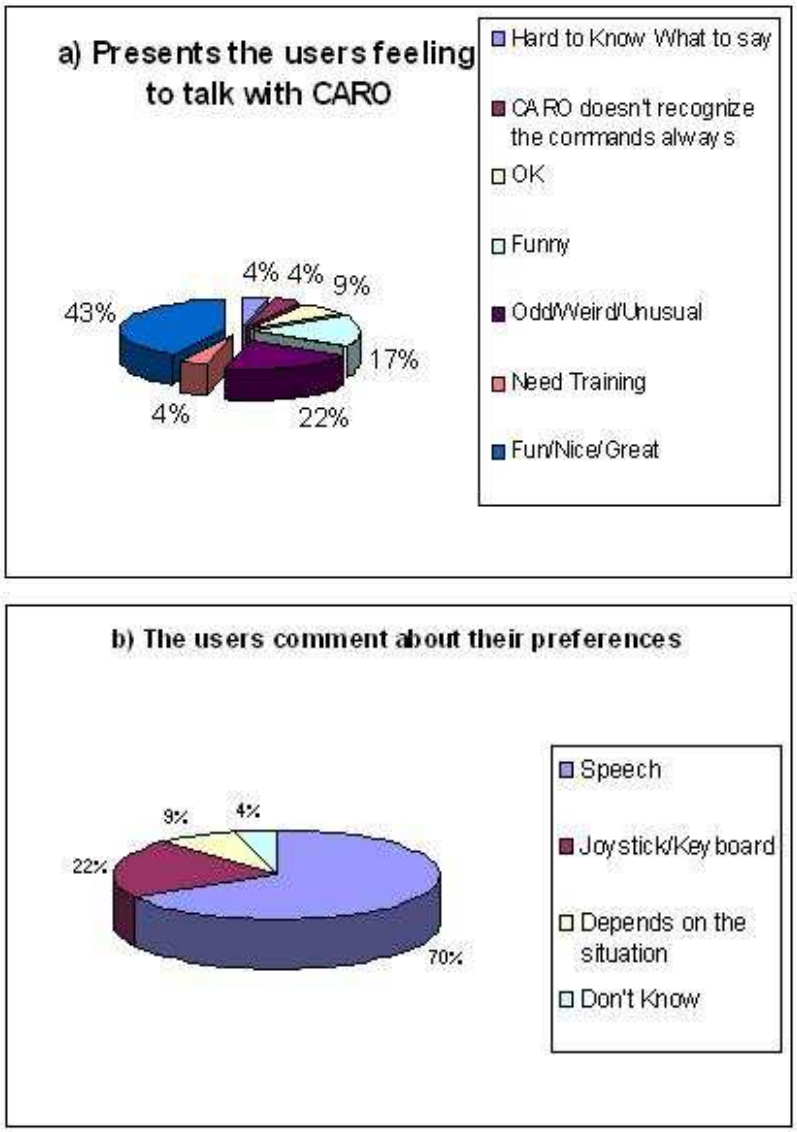

Figure 5.9: The Users comment about their preferences.

Chapter 6

Discussion

The test results give us the facts about our success, problems and limitation to introduce SR system as interface for robotics control. Here we mainly discuss the overall test results, which we have presented in the Chapter 5. This discussion gives the reader an overview of the test results. First we have some discussion about Hardware approach, then the Software approach test results. We also discuss about the achievement at the Technical Fair.

In the Hardware approach we have used VE Module (SR module). From the test results we have found that the VE Module's speaker dependent feature is very sensitive. It's not only sensitive to noise, but also sensitive to voice tone changes and microphone position. We have also found that sentences with three words are not always recognized, because the user has to maintain the even tone at every word in the sentence, when he/she gives any command to the robot. The LEDs are also not a suitable interface to give user feedback, because it engages the users so much. Sometime the users simply miss the feedback.

With the Software approach, we have got better result. Here, we have used the software module named "SpeechStudio" as SR module. We have found some limitation in this SR module; we have to keep the noise at a minimal level, when we use the system. Another observation is that when we are not planning to communicate with the system, we have to mute or switch off the microphone connection, because the surrounding noise can make the system malfunctioning. To avoid the system get hurt or crash with the wall, if the users forget to mute the microphone when he/she isn't using - we have introduced Avoid-obstacle behavior to the robot. Sometimes the system can not response with the user speech, the reason behind that is mainly the noise or the user's speak is not so clear or the user say sometime which is not designed to respond to.

We have also achieved a great experience to present our system in the Stockholm International fair'2005 (Tekniska mässan 2005). It was a technical fair, so people have gathered there to learn about the new technology. We have also found different kind of people participation for our system testing. Our project goal is to make interface for the Service Robot, which will work in the social context; and also the interface should be for the novice user. Almost all of the participants were novice user, so the test results help to know their comments about our system. Another interesting thing is that near

about every age group find the system easy to control.

The noise factor affect our system performance quite a lot, so we find that CARO understands the commands often, but when it understands - it does the act perfectly. From the SR documentation [34] we have found that the noise factor affects SR system performance, which is the main key for user interface of our system. A fair is a gathering of people, so the noise makes the system response - Often not Always.

From the users comment, we have found that the commands are flexible enough to control the CARO.

After evaluating all the usability test-results, we have found that majority of the users have given positive response about CARO, so we can conclude that our system satisfied the users.

Chapter 7

Conclusions

Human-Robot interaction is an important, attractive and challenging area in HRI. The Service Robot popularity gives the researcher more interest to work with user interface for robots to make it more user friendly to the social context. Speech Recognition (SR) technology gives the researcher the opportunity to add Natural language (NL) communication with robot in natural and even way. Also the appearance of the SR interface in the standard software application as a Natural Language (NL) user interface in HCI field for the novices encourages Roboticist to use SR technology for the HRI. Most of the presented projects in SR interface for robotics emphasize on Mobile Autonomous Service Robot [30, 6, 22, 20, 11, 17]. The working domain of the Service Robot is in the society -to help the people in every day's life and so it should be controlled by the human. In the social context, the most popular humans' communication media is Spoken Natural Language, so to communicate with human the SR interface for Human-Robot interaction is coined.

Main target of our project is to add SR capabilities in the Mobile Robot and investigate the use of a natural language (NL) such as English as a user interface for interacting with the Robot. We have implemented the SR interface successfully with hardware Speech Recognition (SR) device as well as Software PC based SR system by using a small Mobile Robot named Khepera. We have done the laboratory test with expert users and the real-time test with novice users. After all the implementation and the testing session, we have gained a lot of experience and also found the problems and limitations when introducing SR system as a user interface to robot. From these achieved experiences, we have reached some conclusions. Our first finding is that the hardware SR device is not as matured as the Software PC based SR system. The hardware SR module does not support the complex grammar sentences, which are normal parts of the spoken natural languages. Another thing is that LED is not suitable interface for the user feedback. After testing the system with the novice users in the technical fair, we have found that SR user interface is a promising aid for interaction with robot. It makes them learn quickly to control the robot. We have also found limitation of the Software PC based SR system; the noise factor affects the SR performance of the SRSP. (Speech Recognition Software Program) and also the robot performance - means the robot does malfunctioning. Another thing is that when the user is not planning to control the robot; he/she should mute the microphone. The SRSP supports complex sentences; this gives us opportunity try complex sentences to control the robot and we have successfully

done this experiment.

7.1 Limitations

In the implementation stage, we have followed the requirement which we set in the beginning. According to these, our system only support English language and also the robot's activities are limited to those mentioned in Table 4.1, 4.2 and 4.3.

7.2 Future work

Our future work will focus on introducing more complex activities and sentence to the system and also introducing the non-speech sound recognition [7], like footsteps (close), footsteps (distant) etc. Another focus area will be to introduce gestures, because gestures are one of the important parts of the Natural Language. Humans normally use gestures such as pointing to an object or a direction with the spoken language, i.e., when the human speaks with another human about a close object or location, they normally point at the object/location by using their fingers. There are also researches going on to introduce speech recognition interface with gestures recognition, this interface called multi-modal communication interface [6].

Chapter 8

Acknowledgements

I would like to thank my supervisor, Thomas Hellström for his valuable insights and comments during my Master thesis project. I could not complete this project-work without the help of a number of people. Even though I can't put everyone's name here. I would specially like to thank Per Lindström, International Student coordinator, and my other courses' teachers, who helped me through out my academic life at Umeå University. I am grateful to my supervisor to give me the opportunity to participate in the Stockholm International fair'2005 (Tekniska mässan 2005) and also thank to my fellow collogues, who have participated and helped me in this technical fair.

References

[1] Register Scienece Editor Abram Katz. Operating room computers obey voice commands. New Haven Register.com. 27 December 2001, http://www.europe.stryker.com/i-suite/de/new_haven_-_yale.pdf (visited 2005-08-15).

[2] Ronald C. Arkin. *BEHAVIOR-BASED ROBOTICS*. The MIT press, Cambridge, Massachusetts, London,UK, 1998.

[3] AT&T Labs-Research. http://www.research.att.com/projects/tts/faq.html #Tech-What (visited 2005-10-30).

[4] Braitenberg Vehicles: Networks on Wheels, http://www.mindspring.com/∼gerken /vehicles (visited 2005-11-24).

[5] Rodney A. Brooks, Cynthia Breazeal, Matthew Marjanovic, Brian Scassellati, and Matthew M. Williamson. The cog project: Building a humanoid robot. *Lecture Notes in Computer Science*, 1562:52–87, 1999. citeseer.ist.psu.edu/brooks99cog.html (visited 2005-10-05).

[6] Guido Bugmann. Effective spoken interfaces to service robots:open problems. In *AISB'05:Social Intelligence and Interaction in Animal, Robots and Agents-SSAISB 2005 Convention*, pages 18–22, Hatfield,UK, April 2005.

[7] Michael Cowling and Renate Sitte. Analysis of speech reconition thechiques for use in a non-speech sound recognition system. http://www.elec.uow.edu.au/staff/wysocki/dspcs/papers/004.pdf (visited 2005-07-11).

[8] Survey of the state of the art in human language technology. Cambridge University Press ISBN 0-521-59277-1, 1996. Sponsored by the National Science Foundation and European Union, Additional support was provided by: Center for Spoken Language Understanding, Oregon Graduate Institute, USA and University of Pisa, Italy, http://www.cslu.ogi.edu/HLTsurvey/ (visited 2005-07-11).

[9] Kerstin Dautenhahn. The aisb'05 convention-social intelligence and interaction in animal, robots and agents. In *AISB'05:Social Intelligence and Interaction in Animal, Robots and Agents-SSAISB 2005 Convention*, pages i–iii, Hatfield,UK, April 2005.

[10] Gregory Dudek and Michael Jenkin. *Computational Principles of Mobile Robotics*. The Press Syndicate of the University of Cambridge, Cambridge, UK, first edition, 2000.

[11] Dominique Estival. Adding lanuage capabilities to a small robot. Technical report, University of Melbourne, Australia, 1998.

[12] Itamar Even-Zohar. A general survey of speech recognition programs, 2004. http://www.tau.ac.il/∼itamarez/sr/survey.htm (visited 2005-08-18).

[13] James L. Fuller. Introduction to robotics. http://www.tvcc.cc/staff/fuller/cs281/chap20/chap20.html (visited 2005-05-20).

[14] Thomas Hellström. Assignment 2 : Odometry and the bug algorithm. http://www.cs.umu.se/kurser/TDBD17/VT05/assignment2.doc (visited 2005-12-03).

[15] Thomas Hellström. Forward kinematics for the khepera robot. http://www.cs.umu.se/kurser/TDBD17/VT05/utdelat/kinematics.pdf (visited 2005-10-20).

[16] John E. Hopcroft, Rajeev Motwani, and Jeffrey D. Ullman. *Introduction to automata theory, languages and computation*. Addison-Wesley, Boston, second edition, 2001.

[17] Helge Hüttenrauch, Anders Green, Michael Norman, Lars Oestreicher, and Kerstin Severinson Eklund. Involving users in the design of a mobile office robot. *Systems, Man and Cybernetics, Part C, IEEE Transactions on*, 34, Issue:2:113 – 124, May 2004. ftp://ftp.nada.kth.se/IPLab/TechReports/IPLab-209.pdf (visited 2005-10-20).

[18] K-Team Corporation, Rue Galile 9 - Y-Parc, 1400 Yverdon, SWITZERLAND Tel: +41 (24) 423 89 50 Fax: +41 (24) 423 89 60. *Khepara Documentation & Software*. http://www.k-team.com/download/khepera.html (visited 2005-11-13).

[19] K-Team Corporation, Rue Galile 9 - Y-Parc, 1400 Yverdon, SWITZERLAND Tel: +41 (24) 423 89 50 Fax: +41 (24) 423 89 60. *Khepara User Manual*. http://www.k-team.com/download/khepera.html (visited 2005-11-13).

[20] A. Ghobakhlou+ Q. Song* N. Kasabov+. Rokel: The interactively learning and navigating robot of the knowledge engineering laboratory at otago. In *ICONIP/ANZIIS/ANNES'99 Workshop*, pages 57–59, Dunedin, New Zealand, November 1999. http://www.aut.ac.nz/resources/research/research_institutes/kedri/downloads/pdf/rokel.pdf (visited 2005-10-01).

[21] Library and Archives CANADA. http://www.collectionscanada.ca/gramophone/m2-3004-e.html (visited 2005-10-30).

[22] Pierre Nugues+ Mathias Haage+, Susanne Schötz*. A prototype robot speech interface with multimodal feedback. In *Proceedings of the 2002 IEEE- Int. Workshop Robot and Human Interactive Communication*, pages 247–252, Berlin Germany, September 2005.

[23] Hossein Motallebipour and August Bering. A spoken dialogue system to control robots. Technical report, Dept. of Computer Science, Lund Institute of Technology, Lund, Sweden, 2003.

[24] Robin R. Murphy. *Introduction to AI ROBOTICS*. The MIT press, Cambridge, Massachusetts, London,UK, 2000.

[25] Oxford English Dictionary, http://www.oed.com/ (visited 2005-10-30).

[26] Oxford Advanced Learnerś Dictionary, http://www.oup.com/elt/catalogue/ teachersites/oald7/?cc=se (visited 2005-10-28).

[27] Julie Payette. Advanced human-computer interface and voice processing applications in space. In *HUMAN LANGUAGE TECHNOLOGY: Proceedings of a Workshop, March 8-11*, pages 416–420, Plainsboro, New Jersey, 1994. Canadian Space Agency, Canadian Astronaut Program, St-Hubert, Quebec, J3Y 8Y9, http://acl.ldc.upenn.edu/H/H94/H94-1083.pdf (visited 2005-10-01).

[28] Proceedings of RO-MAN'03. *From HCI to HRI - Usability Inspection in Multimodal Human - Robot Interactions*, November 2003. San Francisco, CA. http://dns1.mor.itesm.mx/∼robotica/Articulos//Ro-man03.pdf (visited 2005-11-18).

[29] Proceedings of the 29th annual meeting on Association for Computational Linguistics. *The Acquisition and Application of Context Sensitive Grammar for English*, 1991. Berkeley,California. http://delivery.acm.org/10.1145/990000/981360/p122-simmons.pdf?key1=981360&key2= 9896203311&coll=portal&dl=ACM&CFID=37207051&CFTOKEN=53915702 (visited 2005-11-21).

[30] Christian Theobalt+ Johan Bos* Tim Chapman+ Arturo Espinosa-Romero* Mark Fraser+ Gillian Hayes* Ewan Klein+ Tetsushi Oka* Richard Reeve+. Talking to godot: Dialogue with a mobile robot. In *Proceedings of 2002 IEEE/RSJ International Conference on Intelligent Robots and System*, pages 1338–1343, Scotland, UK, 2002. http://www.iccs.informatics.ed.ac.uk/∼ewan/Papers/Theobalt:2002:TGD.pdf (visited 2005-08-28).

[31] SENSORY,INC, 1991 Russell Ave., Santa Clara, CA 95054 Tel: (408) 327-9000 Fax: (408) 727-4748. *Voice Extreme^{T}M Module Speech Recognition Module Data sheet*. http://www.sensoryinc.com/ (visited 2005-05-25).

[32] SENSORY,INC, 1991 Russell Ave., Santa Clara, CA 95054 Tel: (408) 327-9000 Fax: (408) 727-4748. *Voice Extreme^{T}M Toolkit Programmer's Manual With Sensory Speech 6 Technology*. http://www.sensoryinc.com/ (visited 2005-05-25).

[33] SpeechStudio Inc., 3104 NW 123rd Place Portland, OR 97229 Tel: 503 520-9664 Fax: 503 210-0324. *Getting Started*. http://www.speechstudio.com/.

[34] SpeechStudio Inc., 3104 NW 123rd Place Portland, OR 97229 Tel: 503 520-9664 Fax: 503 210-0324. *SpeechStudio Overview*. http://www.speechstudio.com/.

[35] SpeechStudio Inc., 3104 NW 123rd Place Portland, OR 97229 Tel: 503 520-9664 Fax: 503 210-0324. *SpeechStudio-Tutorial for VB6.0-Introduction*. http://www.speechstudio.com/.

[36] UNECE: United Nations Economic Commission for Europe. Press Release ECE/STAT/04/P01, Geneva, 20 October 2004, http://www.unece.org/press/pr2004/04stat_p01e.pdf (visited 2005-08-25).

[37] WordNet - a lexical database for the English language, Cognitive Science Laboratory, Princeton University, 221 Nassau St. Princeton, NJ 08542, New Jersey 08544 USA, http://wordnet.princeton.edu/ (visited 2005-10-28).

Appendix A

Hardware & Software Components

A.1 Hardware Components

A.1.1 Voice $Extreme^{TM}$ (VE) Module

Figure A.1: Voice $Extreme^{TM}$ (VE) Module [31].

Voice $Extreme^{TM}$ (VE) Module a speech recognition products in simplifies design onto a single board. It is a reprogrammable module, which can be programmed and downloaded into the VE Module using the Voice $Extreme^{TM}$ Toolkit. After downloaded the program, the module can to unplug from the Development Board and wired into the final product. This module has 34-pin connector; from these 11 pins are for I/O lines, a power, microphone, speaker, and logic-level RS232 interface. Figure A.1 shows the picture of the Voice $Extreme^{TM}$ (VE) Module; it is the top view of the module. [31]

There are 6 different features in this module; there are - Speaker-independent speech recognition, Speaker-dependent speech recognition and word spotting, High quality speech synthesis and sound effects, Speaker verification, Four-voice music synthesis,

Voice record & playback. [31]

Figure A.2 shows the pins configuration of the Voice $Extreme^{TM}$ (VE) Module. *If an application is stand alone, the two serial I/O pins, P0.0 and P0.1, and the serial port enable, P1.7, may be used for other purposes; however, programs will download via asynchronous serial I/O. Since I/O pins P0.5 and P0.6 are connected to the address bus of the Flash memory, they should not be used under any circumstances.* [31]

Pin	Description	Pin	Description	Pin	Description
1	GND	13	DOWNLOAD	25	P1-0
2	GND	14	RESET	26	P1-1
3	GND	15	VDD	27	P1-2
4	MIC-IN	16	PDN	28	P1-3
5	N.C.	17	P0-0 Serial port input from Host (RCV)	29	P1-4
6	DAC-OUT	18	P0-1	30	P1-5
7	GND	19	P0-2 Serial port output to Host (XMT)	31	P1-6
8	GND	20	P0-3	32	P1-7 Serial port enable, output (0=off, 1=on)
9	AUDIO-RET	21	P0-4	33	GND
10	AUDIO-OUT	22	P0-5 DO NOT USE (Flash address bit A16)	34	PND
11	PWM0	23	P0-6 DO NOT USE (Flash address bit A17)		
12	PWM1	24	P0-7		

Figure A.2: Voice $Extreme^{TM}$ (VE) Module's Pins Configuration [31].

A.1.2 Voice $Extreme^{TM}$ (VE) Development Board

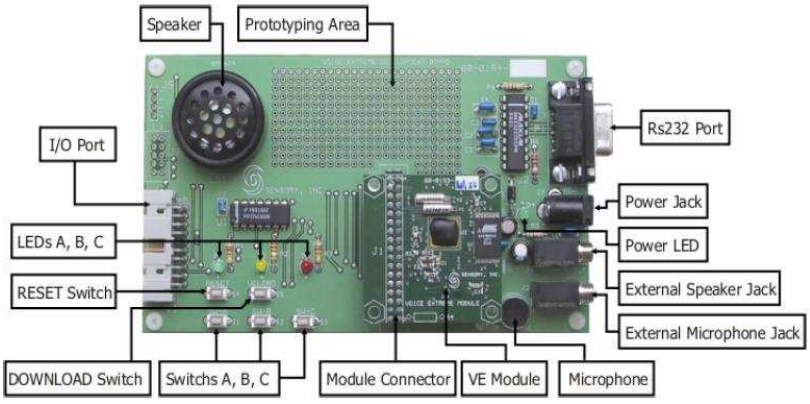

Figure A.3: Voice $Extreme^{TM}$ (VE) Development Board [32].

The Voice $Extreme^{TM}$ Development Board has several features. We have discussed about some important features, such as **Speaker**- there is inboard speaker with fixed

volume and also an output jack for external speaker; the jack will disable the inboard speaker after plug-in the external speaker; this speaker can be used for debugging purpose; **Prototyping Area** - it's a grid of 0.1" through-holes for use by the application developer to add external circuitry; **RS-232 Port** - there is 9 pins connector for connecting to the PC through RS-232 serial cable. **I/O Port** - there are standard 20-pin I/O lines, which can be used from the development board to the target application (See the I/O pins configuration in Figure A.4); **Voice** $Extreme^{TM}$ **Module** - This module is the heart of the system, after downloaded the program to the module; it can be unplugged from the board and wired in the target application; **Microphone** - there is a inboard microphone and also a option to use external microphone through output jack; the microphone is mainly used for debugging or training purpose; **Reset Switch** - it makes the hardware reset of the VE Module; **Download Switch** - it makes the VE Module in a state such that it is waiting for a program to be downloaded from the development PC. **Led 1, 2 and 3** - can be used for development purpose to see the output from the VE module; **Switch A, B and C** - can be used for development purpose. [32]

I/O	Description	I/O	Description
P0.0	Serial port input from Host (RCV)	**P1.0**	Button A, input, strong P/U (0=pressed)
P0.1	Serial port output to Host (XMT)	**P1.1**	Button B, input, strong P/U (0=pressed)
P0.2	Unused, input, weak pull-up	**P1.2**	Button C, input, strong P/U (0=pressed)
P0.3	Unused, input, weak pull-up	**P1.3**	Red LED, output (0=on)
P0.4	Unused, input, weak pull-up	**P1.4**	Unused, input, weak pull-up
P0.5	DO NOT USE (Flash address bit A16)	**P1.5**	Green LED, output (0=on)
P0.6	DO NOT USE (Flash address bit A17)	**P1.6**	Yellow LED, output (0=on)
P0.7	Unused, input, weak pull-up	**P1.7**	Serial port enable, output (0=off, 1=on)

Figure A.4: Voice $Extreme^{TM}$ (VE) Development Board I/O pins configuration [32].

A.1.3 Khepera

Figure A.5: Khepera (a small mobile robot) [18].

Khepera is a small mobile robot for using in research and education purpose. It is

a product from K-team Company. The Khepera robot size in Diameter is: 70 mm; Motion - For motion robot there are 2 DC brushed servo motors with incremental encoders (roughly 12 pulses per mm of robot motion). Perception - there are 8 Infra-red proximity and ambient light sensors with up to 100mm range. The external sensors can be added through General I/O turret (See Figure A.6). The developer can get the development guideline and environment information from the K-team company website (http://www.k-team.com/robots/khepera/index.html). [18]

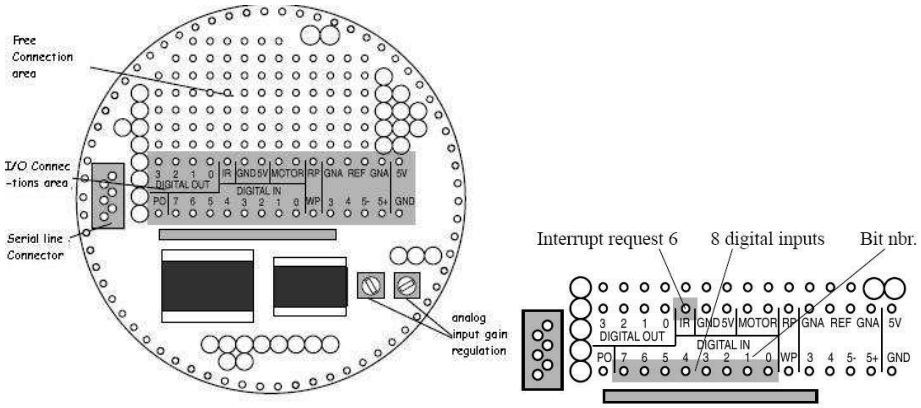

Figure A.6: Overview of the GENERAL I/O TURRET [18].

A.2 Software Components

A.2.1 Voice $Extreme^{TM}$ IDE

To program into VE module, we need to create VE-C applications. VE-C, is very similar to ANSI-standard C and Voice $Extreme^{TM}$ IDE is the development environment to create the VE-C applications. After created the application the developer can download the application with help of VE developer board and RS-232 serial port. The developer need to load the binary file (.VEB) to the VE module. To develop the features application - **Speaker Independent Speech Recognition, Speaker Dependent Speech Recognition, Speaker Verification, Continuous Listening, WordSpot, Record and Play, TouchTones (DTMF), Music** - in VE module through VE-C application the developer need to use different type of data types and functions, which are build-in data type and function of the Voice $Extreme^{TM}$ IDE. Here we have discussed about some of this feature, which is related to our project. [32]

Speaker Independent Speech Recognition : The developer need to make link the program to a WEIGHTS file, which is used to guide the neural-net processing during SI Recognition and also have to use PatGenW function to listen for the pattern and Recog function to try to recognize the pattern in the WEIGHTS set. [32]

Speaker Dependent Speech Recognition : This feature is generally used for a single user speech recognition purpose. Here smaller vocabularies give better recognition results, with the maximum practical size being about 64 words. This technology

needs a training set of templates; and after training, stored them in flash memory and then performing recognition against the trained set. In the training phase, PatGen function is used to generate patterns, TrainSD function is used to average two templates to increase the accuracy of recognition, and PutTemplate and GetTemplate functions are used to transfer templates between temporary and permanent storage. At the recognition phase, PatGen is again used to generate a template and RecogSD function is used to perform the recognition. [32]

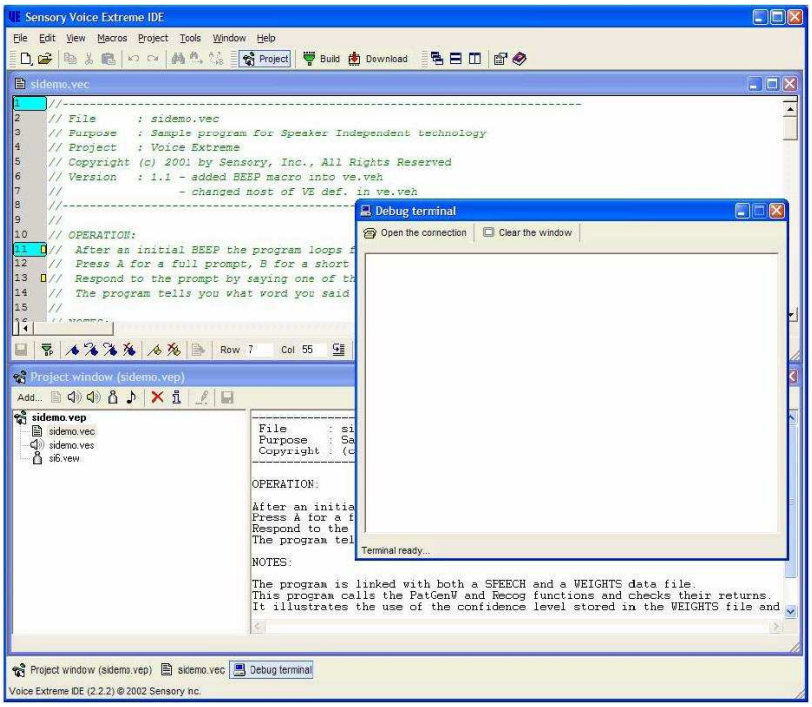
Sensory Voice Extreme IDE
File Edit View Macros Project Tools Window Help
Project Build Download
sidemo.vec
// File : sidemo.vec
// Purpose : Sample program for Speaker Independent technology
// Project : Voice Extreme
// Copyright (c) 2001 by Sensory, Inc., All Rights Reserved
// Version : 1.1 - added BEEP macro into ve.veh
// - changed most of VE def. in ve.veh
// OPERATION:
// After an initial BEEP the program loops
// Press A for a full prompt, B for a short
// Respond to the prompt by saying one of th
// The program tells you what word you said
Row 7 Col 55
Debug terminal
Open the connection
Clear the window
Terminal ready...
Project window (sidemo.vep)
Add...
sidemo.vep
sidemo.vec
sidemo.ves
si6.vew
OPERATION:
NOTES:
The program is linked with both a SPEECH and a WEIGHTS data file.
This program calls the PatGenW and Recog functions and checks their returns.
It illustrates the use of the confidence level stored in the WEIGHTS file and
Project window (sidemo.vep)
sidemo.vec
Debug terminal
Voice Extreme IDE (2.2.2) © 2002 Sensory Inc.

Figure A.7: Voice $Extreme^{TM}$ IDE [32].

Continuous Listening : This feature introduces the capability to listen continuously for a "trigger" word or phrase to be spoken. This technology does not recognize words embedded in speech; the WordSpot technology is available for those applications. CL is generally used to recognize a short command sequence, such as "Place call". Each of these words is recognized individually, with the first word being a "trigger" word and the second word actually causing an "action" to be performed. [32]

A.2.2 SpeechStudio

We have used SpeechStudio to create our project Voice User Interface and most important part of the SpeechStudio is grammar creation. So here we only emphasize on grammar creation through SpeechStudio.

In the SpeechStudio workspace window there is a **Menus, Forms** and **Grammars** folder. The Figure A.8 shows our project application SpeecpStudio workspace window,

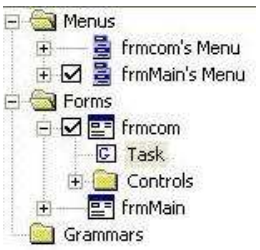

Figure A.8: SpeechStudio workspace window.

if we *Right* click on "frmMain's Menu" under the **Menus** folder or "frmcom" under the **Forms** folder, a popup menu will come and from the popup we can choose Create Grammar to create grammar file for the application. If the developer wants to create grammar for the menu item he/she should *Right* click under the **Menus** folder and if he/she creates for the form's item/object, he/she should *Right* click under the **Forms** folder. So before create grammar developer have to plan a system design that the application can be control through graphical interface, then design for the VUI and modify the GUI according to VUI design. For out project, we have created the GUI using "Option" button and "Text Box" for robotics control and create the grammar with these Form's components. In the Figure A.8 example shows that, there is a "Task.grm", which is a grammar file ("Task" with a G-in-a-box icon appear under "frmcom" in the Forms folder). Figure A.9 also shows the "Task.grm" file after open in the right side of the workspace. The developer can find the grammar syntax in the Start — Programs — SpeechStudio — Tutorials— Introduction /Changing Grammar to create grammar for VUI in an application.

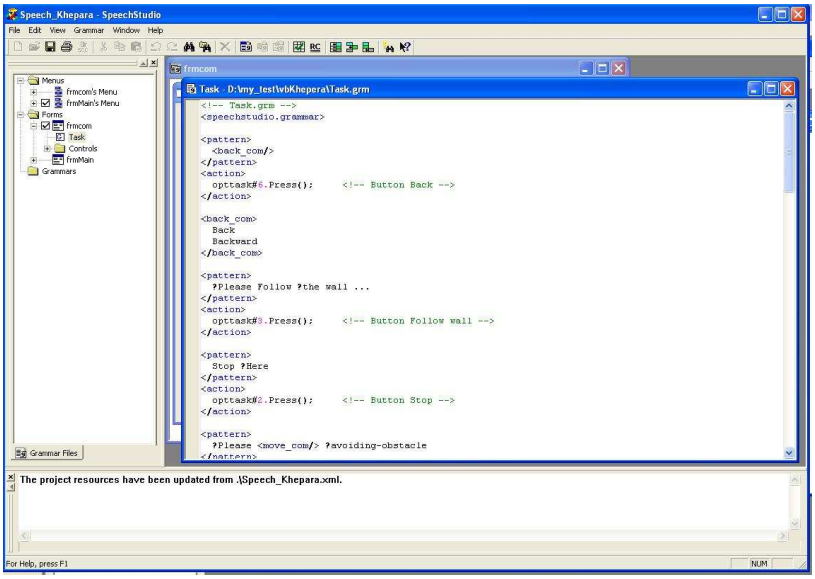

Figure A.9: SpeechStudio grammar creation environment for developer.

Appendix B

Installation guide

Welcome to installation guideline of Voice User Interface (VUI) for Robotic Control. Here we only present the Software approach system's software installation guideline for both the developer and the user. At the user installation, source files are not accessible, only *.exe file available there. We assume that the user follow the Khepera Robot User Manual [19] to connect Khepera with the PC.

B.1 Developer guide

At first, the developer need to install Visual Basic 6.0 (VB6.0) and the SpeechStudio Developer Bundle to get into the source code files of the system. The Visual Basic 6.0 (VB6.0) typical installation is convenient for the system. We present some information about the SpeechStudio Developer Bundle (Speech Recognition Software) below.

B.1.1 Speech Recognition software product installation

You must download and install four packages to complete the entire SpeechStudio Developer Bundle installation. Download the files from the SpeechStudio ftp site:

$< ftp://ftp.speechstudio.com >$ ftp.speechstudio.com

Download these binary files;

Product Name	**File Name**
SpeechStudio	Studio372.msi
SpeechPlayer	SpeechPlayer372.msi
Profile Developer	ProfDev371.msi
Lexicon Developer	LexDeveloper366.msi

Table B.1: The available software products and their file's name in the SpeechStudio Developer Bundle Package.

During installation, you will be prompted for a license key. You will also need a separate user/license key for installing Profile Manager, which is included in Profile Developer.

B.1.2 The Source code files

To get into the source code files developer need to browse **vbKhepera** folder. There, you find **Speech_Khepera.vbp** project file and *Double* click the file to get into the project. After opened the project, you find all the **Forms** and **Modules** in the *Project Explorer* window. You also browse the grammar files from in the VB6.0 by clicking the below icon:

Separately you can browse the grammar files by opening SpeechStudio program from the Manu: **Start— All Programs— SpeechStudio**. The grammar files are in the same directory, where the VB project is. The files are "*.grm" extension.

B.2 User guide

You will find a Setup.exe file to install the system. During installation, you will be prompted for changing directory. The default directory is set in c:\Program files\ Speech_Khepera.

After successfully install the system you can find it:
Start— All Programs— Speech_Khepera— Speech_Khepera.

Click the Speech_Khepera to get start the system

You also need to install SpeechPlayer to activate the SR system. SpeechPlayer is a SpeechStudio's product. You can download free installation file from the SpeechStudio ftp site:

$< ftp://ftp.speechstudio.com >$ ftp.speechstudio.com

Download the binary file:

SpeechPlayer372.msi – "SpeechPlayer"

You don't need a license key to install the SpeechPlayer.

Note:

⋆ If the system gives error -*you do not have a speech engine installed*, then you have install Microsoft SAPI 5 English. You can download free SAPI 5 engine from www.microsoft.com /Speech/download/sdk51 as part of the SAPI 5.1 SDK. [33]

⋆ *You may see a "Server Busy" message box, indicating that SpeechPlayer is still initializing the speech engine; if so, just click "Retry"* [33].

⋆ After starting the system, look at the bottom of the SpeechPlayer window. The lower left window will show status going from "Starting...", to "Not Listening" to "Listening" when the engine is ready. The lower right-hand window is a microphone level meter. If you have a microphone plugged in and working, you should now be able to talk to the system. Try to talk with the system with simple word "move"; it should be work;

you can see the Khepera moves forward and also the system *message window* shows the command. If it is not recognize the word, you should perform a training session through "Profile Manager" to increase the SR performance. You can find **Start— All Programs— SpeechStudio—Tools— Profile Manager**. [33]

Appendix C

User Questionnaire

—— The Candy Robot CARO - User questionnaire ——

Your age: .....

Sex: Male / Female

Current occupation (student or job): .....

1. How do feel to talk with CARO?

.....

2. Did you manage to get a candy out? .....

3. If yes, how long time did it take? .....

4. Did you find it hard to control CARO?
1) It was very easy
2) It was fairly easy
3) It was pretty hard
4) It was very hard

5. Do you find the delay time disturbing?
1) Yes, it takes CARO a very long time to understand what I am saying
2) Yes, but it is not a big problem
3) No, it is ok.

6. Are the commands flexible enough to operate CARO?

.....

7. When you told CARO to do something - did it act like you expected?
1) Always
2) Often

3) Seldom
4) Never

8. If CARO did not do what you told it, what happened?
1) CARO did nothing
2) CARO did something else
3) CARO did the right thing, but not what I intended

9. Would you prefer to control the robot with speech instead of joystick or keyboard?

.....

10. Did you get enough help from the CARO when get it got stuck?

.....

Appendix D

Glossary

CFG - Context Free Grammar
CL - Continuous Listening
GUI - Graphical User Interface
HCI - Human-Computer Interaction
HRI - Human-Robot Interaction
Khepera - a small mobile robot's name
NL - Natural Language
SD - Speaker Dependent
SI - Speaker independent
SR - Speech Recognition
SRHM - Speech Recognition Hardware Module
SRSP - Speech Recognition Software Program
TTS - Text-To-Speech synthesis technology
UI - User Interface
VE Module - Voice $Extreme^{TM}$ (VE) Module
VUI - Voice User Interface